# Universidad Carlos III de Madrid

## Escuela Politécnica Superior


Ingeniería Técnica de Telecomunicaciones
especialidad Sonido e Imagen

# Java Training: Diseño e implementación de una aplicación de aprendizaje Java a través de un dispositivo móvil Android

Autor: Manuel Mercadal Minguijón
Tutores: Rosa Filgueira Vicente y Alejandro Calderón Mateos

# Índice de contenido

# Índice de ilustraciones

# Índice de tablas

# 1. Introducción

## 1.1. Objetivos

El objetivo principal de este proyecto es crear una aplicación para aprender y practicar el lenguaje de programación Java desde un dispositivo móvil.

La disponibilidad actual de plataformas con software libre como Android o iPhone [1] nos ofrece herramientas bastante útiles para el desarrollo de aplicaciones. Esto nos lleva a un mercado bastante interesante con posibilidades reales en la integración de este tipo de dispositivos en las actividades lectivas.

Además, durante este desarrollo, se han establecido una serie de objetivos que detallo a continuación:

- ✓ Introducción e investigación a la plataforma de desarrollo de software para dispositivos móviles Android. Aprender sus conceptos básicos y a manejar su Interfaz de Programación de Aplicaciones (API) y sus Herramientas de Desarrollo de Software (SDK [2]), así como su instalación.
- ✓ Manejo del lenguaje de programación Java y del entorno de desarrollo Eclipse, así como del emulador de Android que incluye en su SDK para poder desarrollar sus aplicaciones.
- ✓ Estudio de la arquitectura cliente-servidor para diseñar la comunicación del dispositivo con una aplicación en un servidor. Desarrollo de componentes Web con Servlets.
- ✓ Conocimiento de entornos de trabajo en Ubuntu, software para implementar máquinas virtuales como VirtualBox.
- ✓ Análisis del actual panorama de las aplicaciones '*Training*' para desarrollar una programa funcional, usable y competitivo.
- ✓ Desarrollo de una aplicación real y útil, enfocada a un usuario que pretenda mejorar sus conocimientos sobre programación en lenguaje Java.
- ✓ Redactar un manual (este documento) que permita una primera aproximación al desarrollo de aplicaciones Android y recabar una documentación básica sobre la

plataforma y su entorno de trabajo.

## 1.2. Estructura del documento

Este documento esta dividido en 9 capítulos que comprenden una incursión creciente en el entorno de trabajo Android.

- En primer lugar, en este capítulo se marcan los objetivos que se van a perseguir a lo largo del desarrollo para orientar y ubicar al lector.
- En el capítulo 2 se expone una introducción a la plataforma Android. Empezando con una explicación contextual de como se gestó Android y como ha ido evolucionando para terminar con una visión mas técnica del entorno, su arquitectura y su interfaz gráfica
- El capítulo 3 pretende ser una primera aproximación para un usuario que no conozca Android pero esta interesado en conocer e instalar el entorno de programación y desarrollar una primera aplicación *Hello World*.
- Una vez ya conocemos el entorno, en el capítulo 4 aborda el desarrollo de la aplicación en cuestión, *Java Training*, incluyendo toda la toma de decisiones sobre las soluciones adoptadas en algunos casos y un desglose de los elementos utilizados en la interfaz.
- El capítulo 5 estructura todas las tecnologías y soluciones que se han necesitado para desarrollar la aplicación.
- En los capítulos 6 y 7 se planifican y presupuesta las labores de análisis y desarrollo de la aplicación para implementar una versión real.
- El capítulo 8 consiste en un análisis de los objetivos cumplidos, las conclusiones y el trabajo de cara al futuro de la aplicación.
- El capítulo 9 contiene dos anexos. El primero es un manual de la aplicación para un usuario final de esta. En el segundo se detalla un glosario de la terminología utilizado a lo largo del presente documento.
- El capítulo 10 recoge la bibliografía utilizada para redactar esta memoria.

# 2. Android

La comunidad de desarrollo móvil está en un punto de inflexión. Los usuarios de móviles exigen más capacidad de elección, más oportunidades de personalizar sus teléfonos, y más funcionalidad. Las operadoras de móvil quieren ofrecer contenido de valor añadido a sus abonados con posibilidad de mercado y los desarrolladores móviles quieren la libertad para desarrollar aplicaciones y así satisfacer con éxito la demanda mínima de los usuarios de estos dispositivos. Por último, los fabricantes de teléfonos móviles quieren una plataforma estable, segura y asequible para sus dispositivos.

Android es un sistema operativo para dispositivos móviles y computadoras basado en el núcleo Linux. Inicialmente fue desarrollado por Google y luego por la *Open Handset Alliance* (liderada por la propia Google). La presentación de la plataforma Android se realizó el 5 de noviembre de 2007 junto con la fundación *Open Handset Alliance*, un consorcio de 48 compañías de hardware, software y telecomunicaciones comprometidas a la promoción de estándares abiertos para dispositivos móviles.

Esta plataforma permite el desarrollo de aplicaciones por terceros (personas ajenas a Google). Los desarrolladores deben escribir código gestionado en lenguaje de programación Java a través de SDK proporcionada por el mismo Google. Una alternativa es el uso de la NDK [3] (*Native Development Kit*) de Google para hacer el desarrollo en C en código fuente.

La mayoría del código fuente de Android ha sido publicado bajo la licencia de software Apache, una licencia de software libre y código fuente abierto. La última versión de Android que Google ha lanzado hasta ahora (el 26 de Octubre del 2009) es la 2.0 (Eclair)


*Ilustración 1: Logo de Android*

## 2.1. Historia de Android

En Julio de 2005, Google adquirió Android, Inc., una pequeña empresa tecnológica, con base en Palo Alto, California. Los co-fundadores de Android que fueron a trabajar a Google incluían a Andy Rubin (co-fundador de Danger), Rich Miner (co-fundador de Wildfire Communications, Inc.), Nick Sears (vicepresidente en T-Mobile), y Chris White (diseño y creo la interfaz de WebTV). En este tiempo, poco se sabia de las funciones de Android, Inc., salvo que desarrollaban software para móviles Así empezaron los rumores de que Google estaba planeando entrar en el mercado de la telefonía móvil, aunque era desconocido el papel que iban a desempeñar.

En Google, el equipo liderado por Rubin desarrolló una plataforma para dispositivos móviles sobre el kernel [5] de Linux, con el objetivo de proveer un sistema flexible y actualizable. Entonces se dijo que Google ya tenía en mente una serie de socios de componentes hardware y software para cooperar.

En Diciembre de 2006, la BBC y The Wall Street Journal publicaron que Google quería desarrollar sus propias aplicaciones y estaban trabajando duro para ello. El resto de medios se hizo eco de estos rumores y llegaron a la conclusión de que Google pretendía desarrollar un dispositivo basado en Google. Las siguientes informaciones señalaban que Google estaba definiendo una serie de especificaciones técnicas, enseñando prototipos a empresas móviles y operadoras de red.

En 2007, InformationWeek publicó que Google había patentado una serie de aplicaciones en el área de telefonía móvil

**Open Handset Alliance**

El 5 de noviembre de 2007, el Open Handset Alliance, un consorcio de varias empresas que incluyen a Texas Instruments, Broadcom Corporation, Google, HTC, Intel, LG, Marvell Technology Group, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel y T-Mobile se dio a conocer con el objetivo de desarrollar estándares abiertos para dispositivos móviles. Junto con la formación de la Alianza Open Handset, la OHA, también dio a conocer su primer producto, Android, una plataforma de dispositivos móviles basada en la versión del kernel de Linux 2.6.

El 9 de diciembre de 2008, se anunció que 14 nuevos miembros se incorporan al proyecto Android, incluyendo: ARM Holdings Plc, Atheros Communications, ASUSTeK Computer Inc., Garmin Ltd., Softbank, Sony Ericsson, Toshiba, y Vodafone Group Plc.

Presidente y CEO de Google, Eric Schmidt, tomó algún tiempo en el comunicado de prensa oficial para disipar todos los rumores y la especulación anterior sobre la existencia de un teléfono stand-alone de Google.

> "Este anuncio es más ambicioso que cualquier "Google Phone" que la prensa ha estado especulando tanto durante las últimas semanas. Nuestra visión es que la potente plataforma que estamos anunciando hoy será motor de miles de modelos distintos de teléfono".
>
> - Eric Schmidt, Google Chairman/CEO

**Licencia**

A la excepción de los breves períodos de actualización, Android esta disponible como código abierto desde el 21 de octubre de 2008. Google abrió el código fuente completo (incluyendo la red y pilas de telefonía) bajo una licencia de Apache.

Con la Licencia Apache, los proveedores son libres de añadir extensiones propietarias, sin tener que presentarlos de vuelta a la comunidad de código abierto.

**Cuota de mercado**

El teléfono Android primero fue lanzado el 22 de octubre de 2008. La compañía de investigación Canalys estima que para el segundo trimestre de 2009, Android tenía una cuota de 2,8% del mercado de *smartphones* [6] en todo el mundo. En el siguiente trimestre, la cuota de mercado Android había crecido hasta el 3,5%.

En octubre de 2009, Gartner Inc., predijo que para el año 2012, Android se convertiría en la segunda plataforma *smartphone* más popular, sólo superada por el sistema operativo Symbian [7] de los teléfonos Nokia, muy popular fuera de los EE.UU. Mientras tanto, BlackBerry [8] bajaría del 2º al 5º lugar, el iPhone se quedaría en 3º lugar, y Windows Mobile [9] de Microsoft se quedaría en el 4º lugar. Taiwan's Market Intelligence & Consulting Institute (CMI) predijo que en 2013, 31.8 millones de teléfonos Android y 126 millones productos

portátiles basados en Android se pondría a la venta.
La empresa de análisis, Flurry, estima que 250.000 teléfonos Motorola Droid se vendieron en los Estados Unidos, durante la primera semana del teléfono en las tiendas.

## 2.2. Arquitectura de Android

Los componentes principales del sistema operativo de Android (cada sección se describe en detalle):

- **Aplicaciones**: las aplicaciones base incluirán un cliente de email, programa de SMS, calendario, mapas, navegador, contactos, y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.
- **Framework de aplicaciones:** Son las APIs para obtener las funciones básicas del móvil y con las cuales programar las aplicaciones. Los desarrolladores tienen acceso completo a los mismos APIs del *framework* usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del *framework*). Éste mismo mecanismo permite que los componentes sean reemplazados por el usuario.

  Los desarrolladores tienen acceso completo a las APIs usadas para desarrollar todo el sistema (Android). Además también tenemos a nuestra disposición las librerías de las aplicaciones de Google (com.google…).

  La lista de servicios aportados por las APIs incluye:

  1. Un amplio conjunto de opciones que se pueden utilizar para construir formularios: listas, cuadros de texto, botones, y hasta un navegador Web empotrable.
  2. Proveedores de contenido que permiten a las aplicaciones acceder a los datos de otras aplicaciones (como los contactos), o para compartir sus propios datos.
  3. Notification Manager, que permite a todas las aplicaciones personalizadas y mostrar las alertas en la barra de estado

4. Activity Manager, que gestiona el ciclo de vida de las aplicaciones y proporciona una navegación hacia atrás.

- **Librerías**: Android incluye un conjunto de librerías utilizadas por diversos componentes del sistema. Escritas en C/C++ [10] son las encargadas de comunicar la HAL [11] de Linux con las APIs y las aplicaciones. Las principales librerías y sus características se enumeran a continuación:
    - Libc. Librerías básicas de C.
    - Surface Manager. Es el encargado de la gestión de las ventanas gráficas que se muestran por pantalla y que forman parte de las diversas aplicaciones y procesos que se ejecuten en el sistema.
    - OpenGL/ES, SGL. Librerías gráficas del sistema. Las librerías openGL son las encargadas del soporte en 3D si el dispositivo soporta aceleración gráfica.
    - SSL. Capa de seguridad de Android.
    - Media Framework. Son las bibliotecas para que el dispositivo soporte multimedia, basadas en las librerías de PacketVideo. Soporte de reproducción y grabación de múltiples formatos audio y video, así como archivos de imagen, (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG…).
    - LibWebCore. Moderno motor de navegación Web.
    - FreeType. Soporte para manejar fuentes de mapas de bits y vectoriales.
    - SQLite. Potente y ligero motor de base de datos relacional a disposición de todas las aplicaciones.
- **Runtime de Android:** El componente principal de esta parte es la máquina virtual Dalvik, que ha sido diseñada específicamente para esta plataforma por Dan Bornstein con contribuciones de otros ingenieros de Google. Está optimizada para requerir poca memoria y está diseñada para permitir ejecutar varias instancias de la máquina virtual simultáneamente, delegando al sistema operativo subyacente el soporte de isolación,

gestión de memoria e hilos.

A menudo Dalvik es nombrada como una máquina virtual Java, pero esto no es estrictamente correcto ya que el bytecode [28] con el que opera no es Java bytecode. Sin embargo, la herramienta dx incluida en el SDK permite transformar los archivos Class de Java compilados por un compilador Java al formato de archivos Dex (formato de bytecode de esta maquina virtual). El nombre de Dalvik fue elegido por Bornstein en honor a Dalvík, un pueblo de Islandia donde vivieron antepasados suyos.

En esta capa también aparece una parte denominada Core Libraries que contiene las clases más básicas para manejar el sistema, por ejemplo el manejo de la I/O.

- **Kernel - Linux**: Android depende de Linux versión 2.6 para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, stack [27] de red, y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto del stack de software.

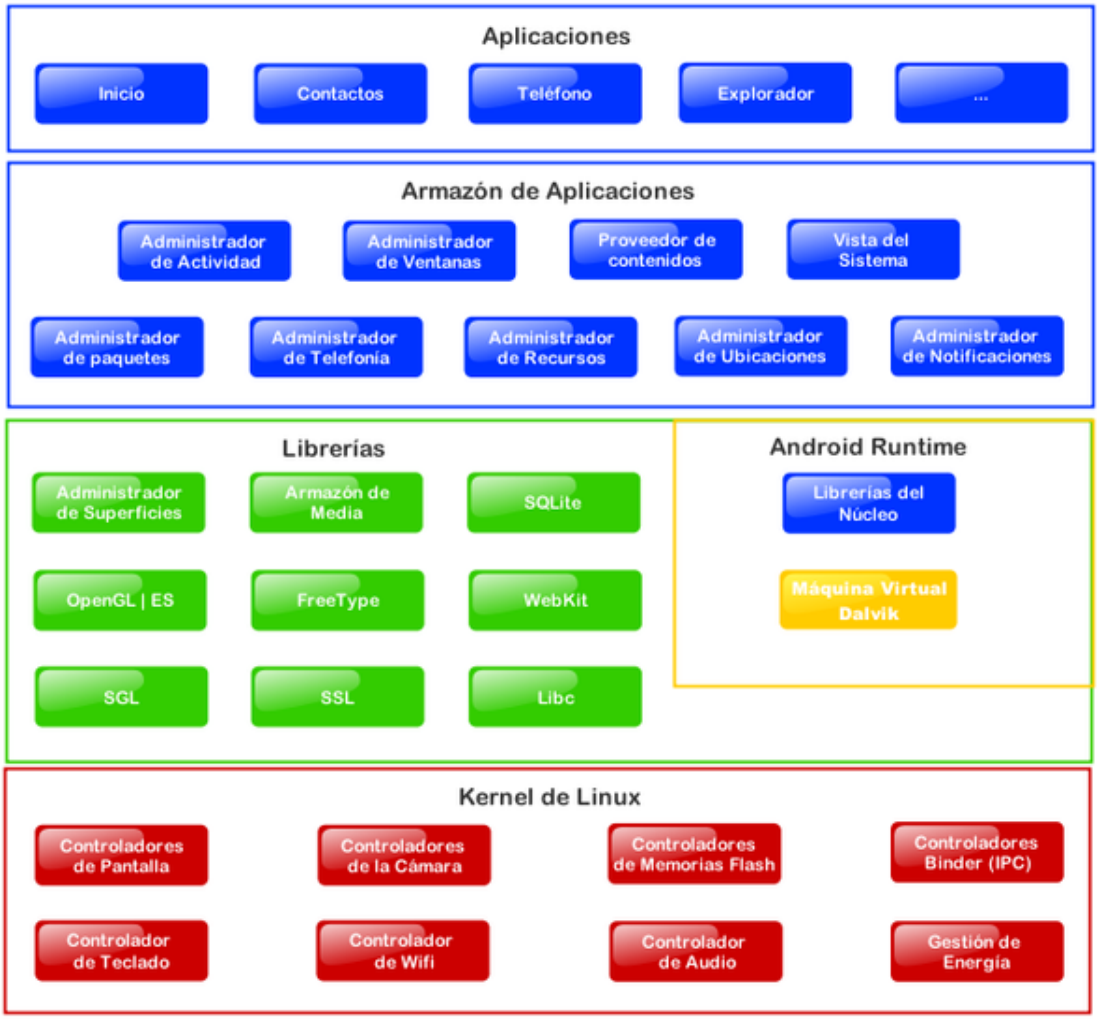



*Ilustración 2: Arquitectura de Android*

## 2.3. Características

| | |
|---|---|
| *Framework* de aplicaciones: | Permite reutilización y reemplazo de componentes. |
| Soporte Java: | Software escrito en Java puede ser compilado para ser ejecutado en la Maquina Virtual Dalvik, que es una implementación de VM diseñada para uso en dispositivos móviles, aunque no es técnicamente un estándar de la máquina virtual de Java. |

| | |
|---|---|
| Navegador integrado: | Basado en el motor de código abierto WebKit. |
| Diseño del Dispositivo: | Gráficos optimizados, con una biblioteca de gráficos 2D; gráficos 3D basado en la especificación OpenGL ES 1.0 (aceleración por hardware opcional). |
| Almacenamiento | SQLite para almacenamiento de datos estructurados. |
| Media | Soporte para medios con formatos comunes de audio, vídeo e imágenes planas (MPEG4 [12], H.264 [13], MP3 [14), OGG [15], AAC [16], AMR [17], JPG [18], PNG [19], GIF [20] |
| Conectividad: | GSM [21], Bluetooth [22], EDGE[23], 3G [24], y WiFi [25] |
| Hardware adicional | Cámara, GPS [26], brújula, Pantalla táctil, y acelerómetro |
| Entorno de desarrollo | Incluye un emulador de dispositivo, herramientas para depurar, perfiles de memoria y rendimiento, y un complemento para el IDE Eclipse. |
| Android Market | Permite que los desarrolladores pongan sus aplicaciones, gratuitas o de pago, en el mercado a través de esta aplicación accesible desde todos los teléfonos con Android. |
| Además de esto, puedes acceder desde tu móvil programando diferentes utilidades y herramientas | |

Una aplicación Android ha de ser diseñada de forma que el usuario la utilice cómodamente, siempre teniendo en cuenta las reducidas dimensiones de la pantalla y la funcionalidad que buscamos.

## 2.4. Jerarquía visual de Android

La unidad funcional básica de una aplicación Android es la "Activity", el cual es un objeto de la clase "android.app.Activity".

Una "Activity" puede hacer muchas cosas, pero por si misma ella no tiene presencia en la pantalla. Para otorgarle a "Activity" presencia en la pantalla y diseñar su interface de usuario, se trabaja con "View" y "ViewGroup", las cuales son las unidades básicas de expresión de la interface de usuario en la plataforma Android.

**Views**: Una view es un objeto cuya clase es android.view.View. Es una estructura de datos cuyas propiedades contienen los datos de la capa y la información específica del área rectangular de la pantalla. Una view tiene: layout, drawing, focus change, scrolling, etc..

La clase view es útil como clase base para los *widgets*, que son unas subclases ya implementadas que dibujan los elementos en la pantalla. Los *widgets* contienen sus propias medidas, pero puedes usarlas para construir tu interfaz más rápidamente. La lista de *widgets* usables incluyen Text, EditText, InputMethod, MovementMethod, Button, RadioButton, CheckBox, y ScrollView.

**Viewgroups:** Un viewgroup es un objeto de la clase android.view.Viewgroup, como su propio nombre indica, un viewgroup es un objeto especial de view cuya función es contener y controlar la lista de views y de otros viewgroups. Los viewgroups te permiten añadir estructuras a la interfaz y acumular complejos elementos en la pantalla que son diseccionados por una sola entidad.

La clase viewgroup es útil como base de la clase layouts, que son subclases implementadas que proveen los tipos más comunes de los layouts de pantalla. Los layouts proporcionan una manera de construir una estructura para una lista de views.

**Árbol estructurado de la interfaz UI:** En la plataforma Android tú defines una Activity del UI usando un árbol de nodos view y viewgroups, como vemos en la imagen de abajo. El árbol puede ser tan simple o complejo como necesites hacerlo, y se puede desarrollar usando los *widgets* [36] y layouts que Android proporciona o creando tus propias views.

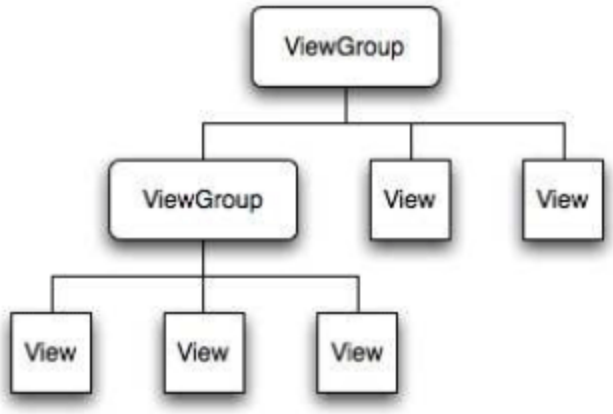



*Ilustración 3: Árbol estructurado de la UI*

Para añadir el árbol a la pantalla, tu Activity llama al método setComtentView() y pasa una referencia al objeto nodo principal. Una vez que el sistema Android ha referenciado el objeto nodo principal ya puede trabajar directamente con el nodo para anular, medir y dibujar el árbol. Cuando tu Activity está activo y recibe el foco el sistema notifica tu Activity y pide al nodo principal medidas y dibuja el árbol. El nodo principal entonces pide que sus nodos hijos se dibujen a sí mismos, a partir de ese momento cada nodo viewgroup del árbol es responsable de pintar sus hijos directos.

Como se ha dicho anteriormente, cada viewgroup es el responsable de tomar medidas sobre el espacio que tienen, preparando a sus hijos y llamando a Draw() por cada hijo que se muestra a si mismo. El hijo hace una petición sobre el tamaño y la localización del padre, pero el objeto padre toma la última decisión sobre el tamaño que cada hijo puede tener.

**LayoutParams:** Cómo un hijo especifica su posición y su tamaño, todos los viewgroup usan como clase anidada una extensión de ViewGroup.LayoutParams. Esta subclase contiene los tipos de propiedades que definen la posición y el tamaño de un hijo, en propiedades apropiadas para la clase de grupo de clases.

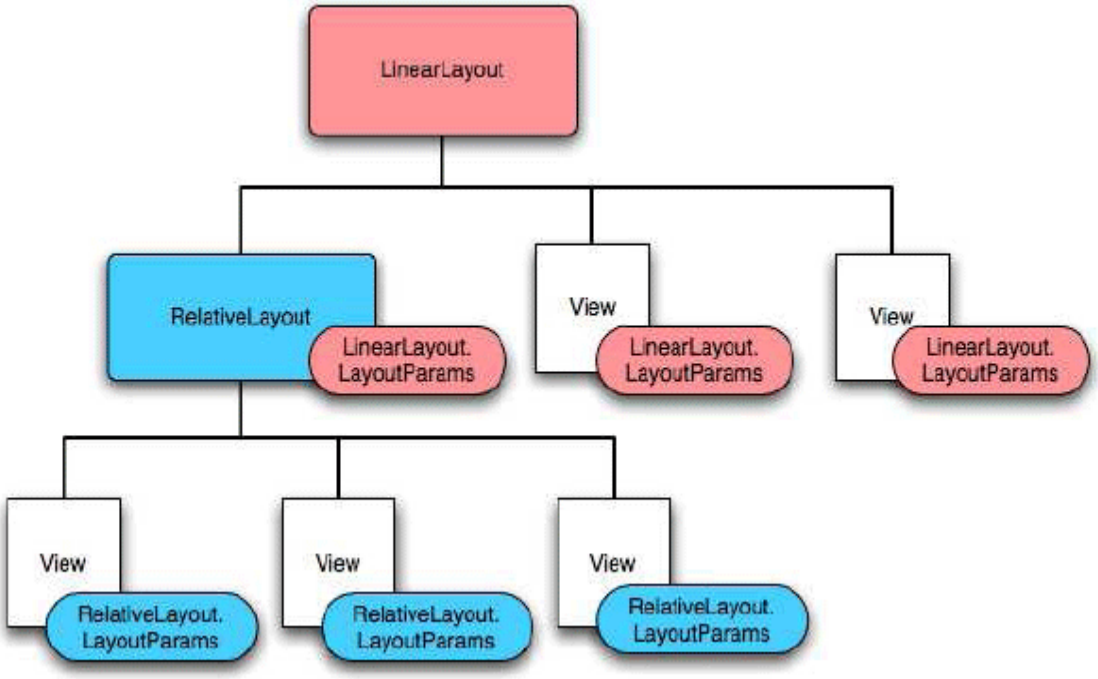



*Ilustración 4: LayaoutParams*

Hay que reseñar que cada subclase LayoutParams tiene su propia sintaxis para cambiar los valores. Cada elemento hijo debe definir unos LayoutParams que sean apropiados para su padre, aunque se podrían definir diferentes LayoutParams para sus hijos.

Todos los viewgroups incluyen anchura y altura. Muchos también incluyen márgenes y bordes. Puedes especificar exactamente la altura y la anchura, hay que pensar que probablemente no se querrá definir esto a menudo. Más a menudo habrá que indicar a la view que tenga las dimensiones del tamaño de su contenedor, o que llegue a ser tan grande como el contenedor le permita.

## 2.5. Actualizaciones del SDK

Android ha sufrido una serie de cambios desde su lanzamiento original. Estas actualizaciones suelen corregir errores y añadir nuevas funciones.

El 30 de abril de 2009, la actualización 1.5 (Cupcake) para Android sale a la luz. Hay varias nuevas características y actualizaciones de la interfaz de usuario incluido en esta versión 1.5:

1.5 (Cupcake)

- Capacidad para grabar y ver vídeos con el modo de cámara de vídeo
- Subir videos a YouTube y fotos a Picasa directamente desde el teléfono
- Un teclado nuevo con la característica de "Autocompletar"
- Soporte Bluetooth A2DP
- Capacidad para conectarse automáticamente a un auricular Bluetooth a una distancia determinada
- Nuevos *widgets* y carpetas para el escritorio.
- Ampliación de la capacidad de copiar y pegar para incluir páginas Web

El 15 de Septiembre de 2009, aparece la SDK 1.6 (Donut). Incluye las siguientes actualizaciones:

1.6 (Donut)

- Una experiencia Android Market mejorada.
- Integra una cámara, una videocámara y la interfaz de la galería
- Galería ahora permite a los usuarios seleccionar varias fotos para su eliminación.
- Actualización Voice Search, con una respuesta más rápida y más una integración más profunda con las aplicaciones nativas, incluyendo la capacidad para marcar los contactos.
- Actualización experiencia de búsqueda para permitir la búsqueda de marcadores, historial, contactos y la Web desde la

pantalla de inicio.

- Actualizado el apoyo de la tecnología de CDMA / EVDO [29], 802.1x [30],VPN [31], los gestos, y un motor texto-a-voz
- Mejoras en la velocidad de búsqueda, la cámara.

El 26 de octubre de 2009, la SDK 2.0 (Eclair) salió a la luz. Los cambios son los siguientes:

2.0 (Eclair)

- Velocidad de hardware optimizado
- Soporte para mas tamaños y resoluciones de pantalla
- Nueva interfaz de usuario.
- Nueva interfaz del navegador y soporte HTML 5 [35]
- Nuevas listas contacto
- Mejor ratio blanco / negro para los fondos
- Mejoras en Google Maps 3.1.2
- Soporte en Microsoft Exchange
- Zoom digital
- Mejora del teclado virtual
- Bluetooth 2.1

La próxima versión del SDK se llamara Flan.

## 2.6. Restricciones e inconvenientes

- Android utiliza Linux como su núcleo, pero según Google, no es una distribución Linux convencional, no tiene un sistema nativo X Window [37], ni tampoco utiliza todo el conjunto de las bibliotecas estándar de GNU tales como su sistema de bibliotecas del sistema (Biblioteca de C de GNU ). Esta modificación específica dificulta la reutilización

de las aplicaciones existentes de Linux o en las bibliotecas en Android.

- Android no utiliza los estándares de Java establecidos, es decir, Java SE y ME [38]. Esto evita la compatibilidad entre las aplicaciones Java se ha creado para esas plataformas y los de la plataforma Android. Android reutiliza la sintaxis del lenguaje Java, pero no proporciona bibliotecas completas de clases y API incluidas en Java SE o ME.
- Debido a problemas de seguridad potenciales, Android no permite instalar, ni ejecutar aplicaciones desde una tarjeta SD. Los dispositivos actuales de Android, como el HTC Dream y Magic han limitado memoria interna y muchos usuarios se quejan por esta falta de funcionalidad. Sin embargo, existen varias modificaciones, sin soporte oficial, que ofrecen al usuario esta capacidad.
- ARM Holdings y RealNetworks se han mostrado escépticos sobre que Android acaparara una importante cuota de mercado como sistema operativo para netbook [40].
- La capacidad de respuesta puede ser pobre debido a las limitaciones de la gestión de memoria automática Dalvik's.
- Los desarrolladores han informado de que es difícil mantener las aplicaciones en diferentes versiones de Android, a causa de diversos problemas de compatibilidad entre las versiones 1.5 y 1.6. Sin embargo, esto sólo fue en algunos casos raros, en concreto, durante el concurso ADC2 [39].

# 3. Primera aplicación en Android

Como ya se ha marcado en los objetivos, puede ser interesante que este documento no solo detalle el diseño y desarrollo de la aplicación *Java Training*, sino que también sirva como manual de consulta y documento para explicar las nociones básicas para poder realizar una aplicación pequeña de forma sencilla. A continuación se va a detallar los pasos necesarios para realizar una aplicación que se encargue de mostrar el clásico "Hola Mundo" .

## 3.1. Instalación

A la hora de desarrollar podemos utilizar diferentes entornos IDE [32], hasta el momento la mejor combinación que se puede recomendar es Eclipse + Android Development Tools (ADT [33]). Éste último componente es el plug-in que se encarga de añadir a Eclipse las funcionalidades necesarias para el desarrollo de aplicaciones Android. Si necesitas descargar Eclipse puedes hacerlo desde aquí:

http://www.eclipse.org/

### 3.1.1. SDK (kit de desarrollo de software )

Antes de instalarlo se descargar de la página oficial de Google Developers:

http://developer.android.com/sdk/

Una vez descargado el archivo, se descomprime una carpeta. Para poder empezar a utilizarlo sólo queda configurar las rutas y añadir el plug-in para Eclipse.

- En Linux, se edita el archivo ~/.bash_profile o ~/.bashrc. Se busca una línea que establece la variable de entorno PATH y se añade la ruta completa del directorio tools/ a la misma. Si no aparece una línea de ajuste de la ruta, se puede añadir una:

  ```
  export PATH=${PATH}:<directorio_sdk>/tools
  ```

- En un Mac, se busca el archivo .bash_profile en el directorio home y se hace lo mismo que en Linux. Se puede crear el archivo .bash_profile si todavía no se tiene uno configurado en la máquina.

- En Windows, botón derecho en Mi PC, se selecciona Propiedades. En la pestaña de Avanzado se hace pulsa en Variables de Sistema, y en el diálogo que aparece se hace doble-click en Path (debajo de Variables del Sistema). Se añade el path absoluto al directorio tools/.

Cabe destacar que si en un futuro se quiere actualizar la versión del SDK hay que volver a configurar el path al directorio tools del nuevo SDK y substituirlo por el anterior.

### 3.1.2. Plug-in ADT para Eclipse

**Eclipse 3.3 (Europa)**

1. Se arranca Eclipse y se selecciona Help >Software Updates >Find and Install….
2. En el diálogo que aparece, se selecciona Search for new features to install y se pulsa en el botón Next.
3. Pulsa New Remote Site.
4. En el cuadro de diálogo que aparece, se introduce un nombre para el sitio remoto (Ej. "Android Plugin") y se introduce la URL:

   http://dl-ssl.google.com/android/eclipse/

   Pulsa en OK.
5. Ahora se debe ver el nuevo sitio añadido a la lista de búsqueda (y comprobado). Pulsa en el botón Finish.
6. En el siguiente cuadro de diálogo Resultados de la búsqueda, se selecciona

**Eclipse 3.4 (Ganymede)**

1. Inicio de Eclipse y selecciona Help > Software Updates….
2. En el cuadro de diálogo que aparece, se pulsa en la pestaña de Available Software.
3. Pulsa en Add Site…
4. Se introduce la URL:

   http://dl-ssl.google.com/android/eclipse/

   Pulsa en OK.
5. De vuelta en Available Software, se debería ver el plug-in de la lista de la URL, con "Developer Tools" anidado dentro de él. Se selecciona la casilla que aparece junto a "Developer Tools" y se pulsa en el botón Install…
6. En la ventana de instalación posterior,

la casilla de verificación para el "Android Plug-in". Esto selecciona las herramientas: "Android DDMS" y "Herramientas de Desarrollo Android". Pulsa en el botón Next.

7. Leer y aceptar la licencia y pulsa en el botón Next.
8. En la siguiente ventana de instalación, se pulsa en el botón Finish.
9. El plug-in ADT no está firmado digitalmente. Aceptar la instalación de todos modos y pulsa en el botón Install All.
10. Reinicio de Eclipse.

"Android DDMS" y "Android Development Tools" deberían estar marcados. Pulsa en el botón Next.

7. Leer y aceptar la licencia y pulsa en el botón Finish.
8. Reinicio de Eclipse.

Ahora se modifican las preferencias de Eclipse para apuntar al directorio del Android SDK:

1. Seleccionar Window > Preferences… para abrir el panel de Preferencias (Mac: Eclipse > Preferences).
2. Seleccionar Android en el panel izquierdo.
3. En la parte de SDK Location, se pulsa en Browse… y buscar el directorio del SDK.
4. Se pulsa en Apply y OK.

## 3.2. Hello World

1. Lo primero que tenemos que hacer es crear un nuevo Proyecto.

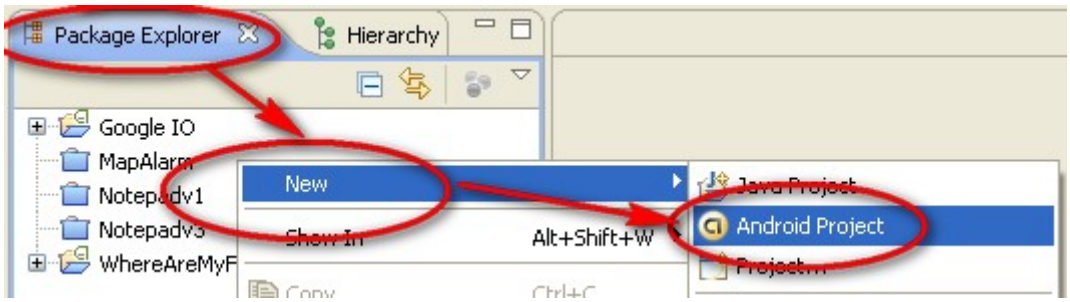



*Ilustración 5: Proyecto de Android nuevo*

2. Se rellena el formulario que aparece. A continuación, una pequeña descripción de cada campo:

   - Project Name

     Nombre del Proyecto de Eclipse – nombre del directorio que contendrá los archivos del proyecto.

   - Application Name

     Título para tu aplicación – es el nombre que aparecerá en los menús del dispositivo Android

   - Package Name

     Es el package namespace (sigue las mismas reglas que los paquetes Java) donde residirá el código fuente. La Activity principal se creará ahí también. El nombre del package debe ser único entre todos los package instalados en el sistema Android; por esa razón, es muy importante utilizar un estándar para el paquete de aplicaciones.

- Create Activity

  Este es el nombre de la clase que se creará. Será una subclase de la clase Activity, una Activity es simplemente una clase que puede ejecutarse y realizar acciones. Se puede crear una interfaz de usuario si lo desea, pero no es necesario. Como se indica en la casilla de verificación, esto es opcional, sino una actividad casi siempre se utiliza como base para una solicitud.

- Min SDK Version

  Este valor especifica el nivel mínimo requerido por la API de su aplicación. El nivel mínimo de la API tiene que coincidir con el nivel de la API del dispositivo virtual que utilicemos (más adelante veremos como crear uno). Si una aplicación necesita un API de nivel superior al nivel del dispositivo, la aplicación no se instalará.

3. Una vez creado el proyecto esta es la estructura de la aplicación:

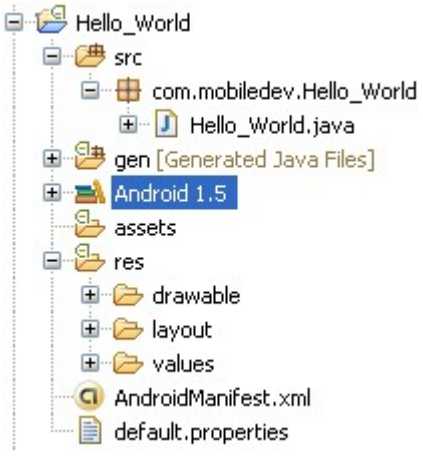



*Ilustración 6: Estructura de la aplicación*

4. El próximo paso es crear un dispositivo virtual donde poder ejecutar las aplicaciones que se desarrollan.

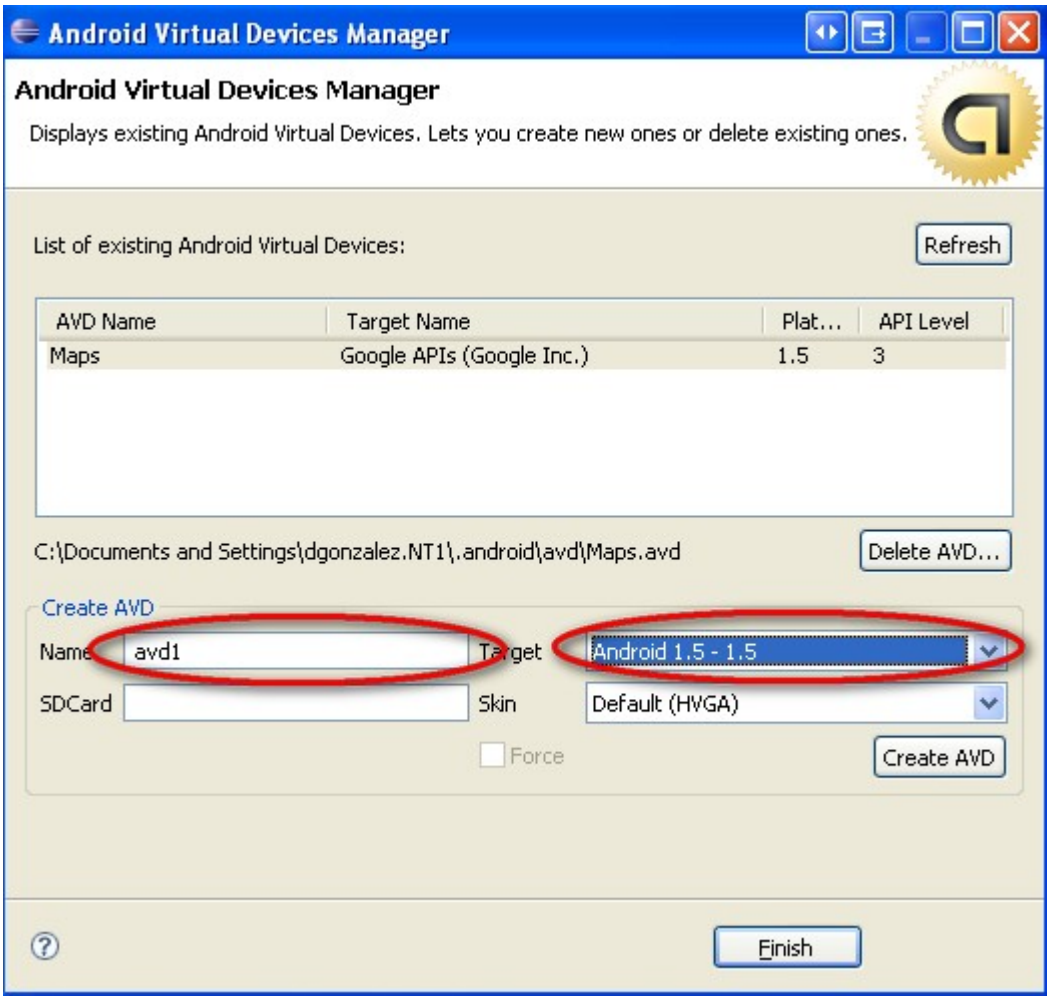



*Ilustración 7: Crear AVD*

5. Ahora si, ya se puede programar la aplicación Java.

Se modifica el método onCreate para que muestre un texto por pantalla. Es lo primero que se ejecuta de una Activity.

La clase  debería quedar de la siguiente manera:

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Hello_World extends Activity {
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            //Para mostrar un texto utilizamos un TextView
            TextView tv = new TextView(this);
            //Se añade de texto al nuevo control
            tv.setText("Hello World - mi primera aplicación" );
            //Muestra el TextView
            setContentView(tv);
      }
}
```

6. Lo único que queda es ejecutar el código y ver el resultado en el emulador. El botón de Run Configurations se encuentra en el menú de Eclipse y al pulsar aparece lo siguiente:

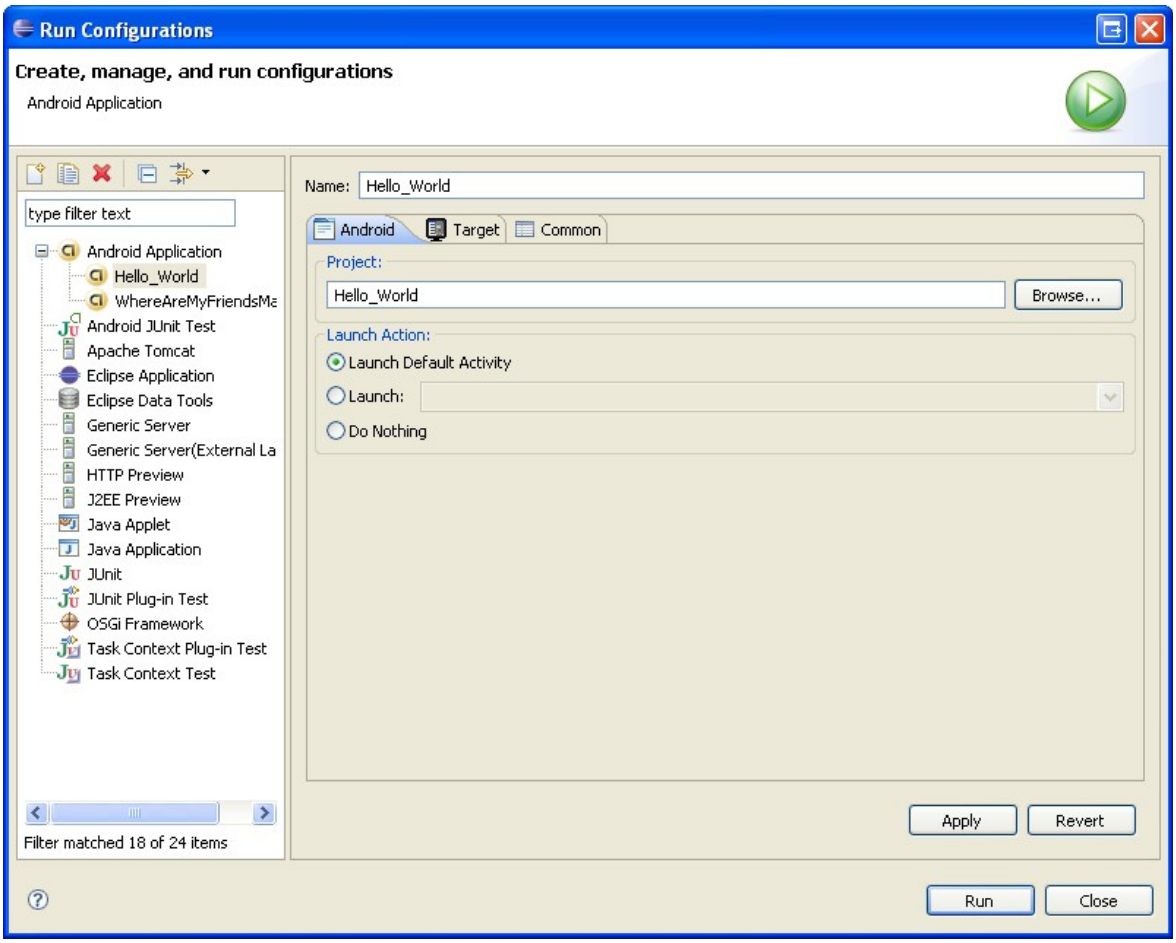
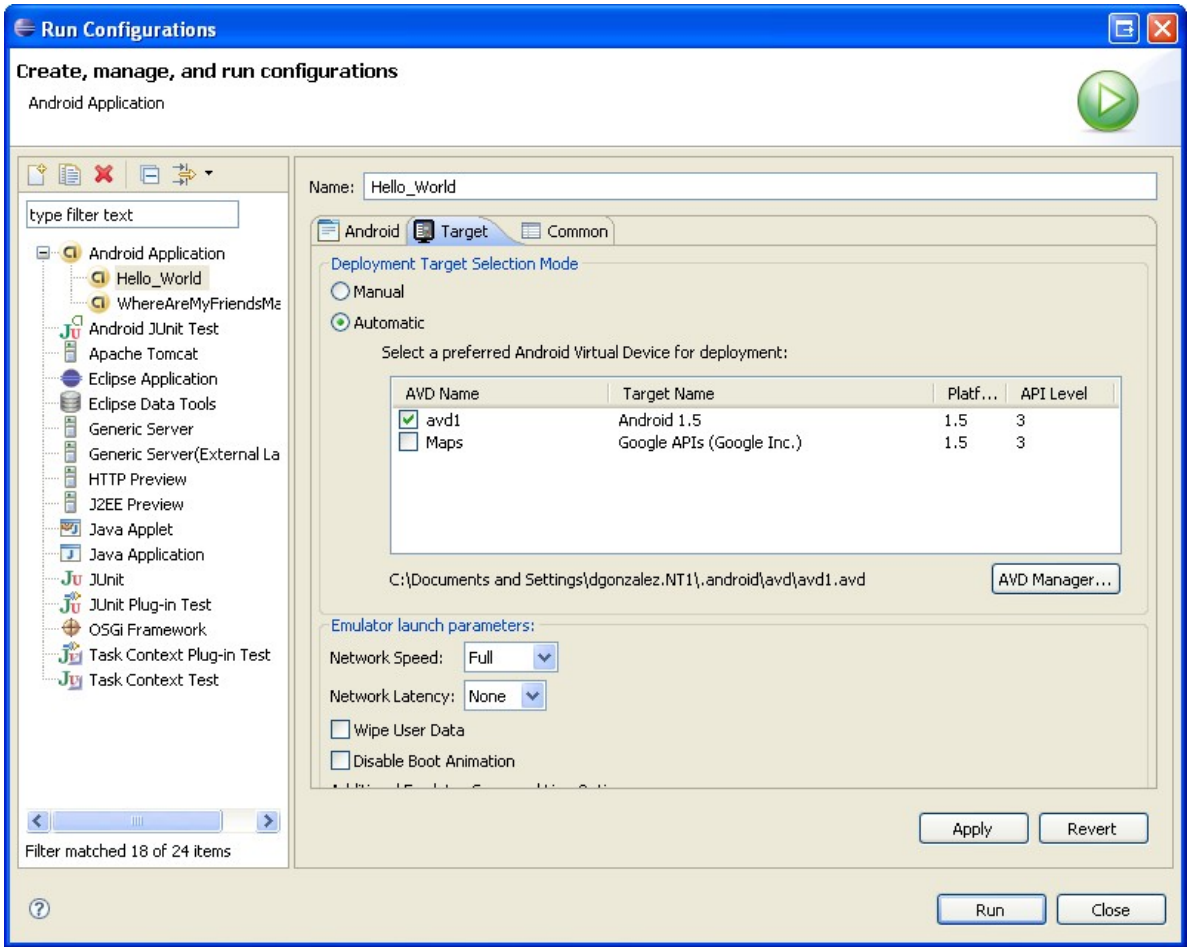



*Ilustración 8: Run configurations*

En la pestaña de target se selecciona el emulador que se ha creado anteriormente.



7. Por ultimo, ejecutamos la aplicación.


*Ilustración 9: Vista de la aplicación ejecutada en el emulador*

# 4. Diseño de la aplicación Java Training

## 4.1. Introducción

La aplicación *Java Training* va a constar de cuatro módulos, incluyendo el menú inicio. Cada uno tiene una funcionalidad y una interfaz distinta que iremos explicando a lo largo de este capítulo:

- menú inicio
- **lecciones**
- **desarrollo**
- ayuda

Los dos más interesantes (tanto a nivel técnico como a nivel de contenidos) son lecciones y desarrollo.

## 4.2. Interfaz de la aplicación

### 4.2.1. Menú inicio

Debido a que el diseño de una pantalla a través de código puede resultar muy complejo, Android soporta una sintaxis XML [34] para diseñar pantallas. Android define un gran número de elementos especialmente hechos para la plataforma, cada uno de ellos representando una subclase específica de Android View. Se puede diseñar una pantalla de la misma forma que se diseña en HTML, es decir, como una serie de etiquetas anidadas y guardadas en un archivo XML dentro del directorio "res/layout/" de la aplicación.

Cada etiqueta describe un único elemento "`android.view.View`", pero este elemento puede ser un simple elemento visual o un elemento de esquema que contiene una colección de objetos hijos (una pantalla o una porción de ella).

Android tiende a dibujar los elementos en el orden en el cual aparecen en el documento XML. Por lo tanto, si hay elementos que se sobreponen, el último en el archivo será probablemente el que aparezca sobre cualquier otro que esté previamente declarado en su mismo espacio.

En el caso de la aplicación *Java Training*, la pantalla menú es también la portada de bienvenida, así que se han utilizado una serie de elementos de la GUI (interfaz gráfica) que permiten utilizar imágenes en vez de solo texto, evitando así que resulte demasiado aséptica o poco atractiva.

Cabecera:

```
<ImageView
    android:id="@+id/ImageView01"
    android:src="@drawable/titulo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</ImageView>
```

Baton imagen:

```
<ImageButton  android:id="@+id/botonlecciones"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/lecciones"
    android:nextFocusDown="@+id/botondesarrollo"
    android:baselineAlignBottom="true"
    android:scaleType="fitStart"/>
```

Cuando Android compila una aplicación, compila cada archivo en un "`android.view.View`" recurso que se puede cargar en el código llamando al método "`setContentView (R.layout.layout_file_name)`" dentro de la implementación que se haga del método "`onCreate()`" de la "`Activity`".

Cada archivo XML esta formado por etiquetas que corresponden a clases GUI de Android. Estas etiquetas tienen atributos que en su mayoría corresponden a métodos en esa clase.

Así pues, en `javatraining.java`, se asigna una funcionalidad a cada botón para que accedan a otra actividad.

```
ImageButton b1= (ImageButton) findViewById(R.id.botonlecciones);
        b1.setOnClickListener(GoToLecciones);

private OnClickListener GoToLecciones = new OnClickListener(){

            public void onClick(View arg0) {
            Intent intent = new Intent();
            intent.setClass(JavaTraining.this, menulecciones.class);
            startActivity(intent);
            finish();
            }
    };
```

### 4.2.2. Lecciones

En este modulo se procura estructurar y organizar los contenidos de una forma que la búsqueda sea fácil. Hay que potenciar la accesibilidad y la usabilidad, así que se deberían contemplar dos modelos de organización:

Por medio de la *jerarquización* identificamos el grado de importancia de ciertos contenidos y la derivación subordinada de otros.

La organización *vertical* se refiera a la ordenación secuencial de los contenidos a través del tiempo, de manera que lo enseñado inicialmente o en etapas previas sea la base y fundamento de lo revisado posteriormente.

Los contenidos se estructuran jerárquicamente, distribuidos en índices-menú bien descritos que faciliten la navegación, pues, en última instancia, también hay que considerar la posibilidad de un seguimiento no-ordenado de las lecciones, como soporte y/o ayuda.

Una vez dentro de la lección, se aprovechan las aplicaciones ya existentes en el sistema operativo Android para visionar las páginas del temario. En este caso, todavía no existe un visor de documentos pdf para Android. Esto me obligó a considerar otras opciones.

La más apropiada sería mostrar las páginas de la lección como imágenes png, que mantiene un excelente ratio de compresión sin perdida.

Para el índice (`menulecciones.java`), la clase `ListActivity` adapta un array de elementos y lo muestra como elemento gráfico, un menú en vertical. Cada elemento es seleccionable y con unos 'listeners' se detecta el pulsado para acceder a la lección correspondiente

```
public void onListItemClick(ListView l, View v, int position, long id)
```


*Ilustración 10: Índice del temario en Lecciones*

Para las lecciones (`lecciones.java`) he desarrollado un visor de imágenes con un elemento `ImageSwitcher` que ocupa toda la pantalla del móvil

```
ImageView i = new ImageView(this);
      i.setBackgroundColor(0xFF000000);
      i.setScaleType(ImageView.ScaleType.FIT_CENTER);
      i.setLayoutParams(new
ImageSwitcher.LayoutParams(LayoutParams.FILL_PARENT,
            LayoutParams.FILL_PARENT));
```

y una galería `Gallery` en la parte inferior donde se muestren todas las páginas y se pueda navegar sobre ellas.

```
Gallery g = (Gallery) findViewById(R.id.gallery);
        g.setAdapter(new ImageAdapter(this));
        g.setOnItemSelectedListener(this);
```


*Ilustración 11: Visor de lecciones*

En el diseño inicial de la aplicación se establecía un control zoom para este visor de imágenes Este control, que consta de dos botones (alejar y acercar), nació con las primeras versiones del sdk de Android.

Es posible que con la llegada de la segunda versión del Sistema Operativo Android

(Android 2.0), se integre el soporte para realizar acciones multitáctiles en las pantallas de estos equipos. Y esto incluye usar el gesto multitáctil de pellizcar para hacer zoom.

Si el equipo detrás del desarrollo de Android decide implementar esta funcionalidad en Android, agregaría interesantes nuevas características en los equipos. Por lo pronto tendríamos un serio rival del iPhone, que basa gran parte de su éxito en sus capacidades multitáctiles.

En ocasiones, las continuas actualizaciones del sdk y las diferencias entre el emulador y el software real implantado en los móviles, han supuesto un contratiempo. La interoperabilidad entre aplicaciones es en ocasiones muy útil (en mi caso, el visor de pdf). Este punto lo desarrollare mas adelante.

### 4.2.3. Desarrollo

La función de desarrollo (`desarrollo`) pretende emular una consola basada en un editor de texto que permita escribir código y compilarlo, obteniendo la salida estructurada en una pantalla aparte.

En el XML (`desarrollo.xml`) definimos dos bloques que puedan contener texto. Uno hace las funciones del input y el otro recoge el output.

En el primer caso, el elemento gráfico debe ser editable, que permita la introducción de texto por parte del usuario.

```
<EditText android:id="@+id/edittext"
      android:layout_width="fill_parent"
      android:layout_height="200px"
      android:gravity="top|left"
      android:textColor="@color/solid_blue"
      android:textSize="10px">
</EditText>
```

A continuación, un botón “compilar” cuya estructura técnica explicaremos mas adelante.

```
<Button android:id="@+id/compilar"
      android:layout_width="160px"
      android:text="@string/compilar"
      android:layout_height="wrap_content">
</Button>
```

Y por último un campo de texto que recoge la respuesta del compilador, que se encuentra en un servidor (con la información de haber compilado el archivo .java). En este caso, hemos contemplado tres posibles soluciones:

- `android.webkit.WebView`
- `android.widget.EditText`
- `android.widget.TextView`

Cualquiera de los tres casos hubiese sido valido para mostrar los resultados de la compilación por pantalla. El problema es que esta 'salida' es un String que no mantiene el formato tal y como se mostraría en la consola de Linux. Por ello, la aplicación Web que hemos diseñado para gestionar la entrada, encapsula la salida en un HTML con etiquetas de salto de línea <br/> para que mantenga el formato.

Esta cadena de texto se imprime en un `WebView`, que es un elemento para visualizar paginas Web y así mantenemos la estructura tal y como se vería en la consola.

```
text.loadData(aux, "text/html", "utf-8");
```

siendo `text` el elemento `WebView` y `aux` el String de salida.

```
<WebView android:id="@+id/compilado"
      android:scrollbars="vertical"
      android:layout_width="fill_parent"
      android:layout_height="200px"
      android:gravity="top|left">
</WebView>
```


*Ilustración 12: Antes de compilar*

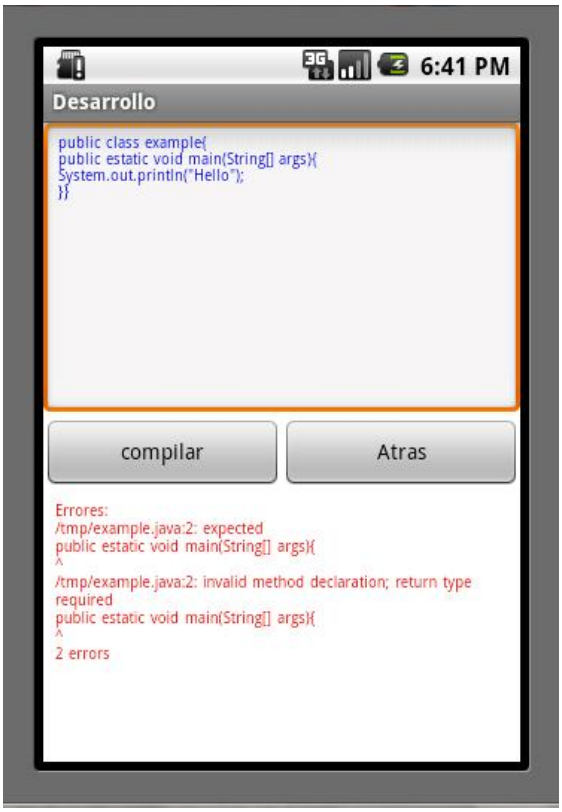



*Ilustración 13: Después de compilar*

A continuación se desarrolla el proceso para desplegar la comunicación entre esta funcionalidad de la aplicación y el servidor.

## 4.3. Arquitectura cliente-servidor

La aplicación consta de dos partes:

- cliente
- servidor

La parte de cliente esta localizada en el programa *Java Training* del terminal Android, en el apartado de `desarrollo` que realiza peticiones a la aplicación WEB en el servidor y este le da respuesta. Es decir, el servidor esta esperando para recibir una petición (escucha) y dar respuesta. Mientras, el cliente envía peticiones pero no escucha.

*Ilustración 14: Arquitectura cliente - servidor*

Cuando se habla de aplicaciones para Web comúnmente escuchamos hablar de Java, y con ello sus aplicaciones mas conocidas, los Applets, que son programas que se pueden cargar a través de una red y que se ejecutan de igual forma en cualquier plataforma, todo ello gracias a las potentes características de Java. Hasta hace poco, Java se utilizaba básicamente para dotar a las páginas WEB de una mayor interactividad mediante los Applets, y por tanto solo actuaba sobre el lado cliente. Pero el lado servidor también puede beneficiarse de todas las ventajas que ofrece Java, gracias a los Servlets.

Los Servlets se diferencian de los Applets básicamente en que se ejecutan en el servidor y en que no presentan ningún tipo de interfaz gráfica puesto que se encargan de hacer el trabajo oculto, un aspecto interesante por lo que muchos programadores que hasta ahora utilizaban CGIs, están utilizando Servlets. De hecho, los CGIs eran el único medio de proporcionar interacción entre el cliente y el servidor.

La API Servlet, usada para escribir servlets, no incluye nada acerca de cómo son cargados los servlets, ni el ambiente en el cual corren los servlets, ni el protocolo usado para transmitir los datos del usuario. Esto permite a los servlets poder ser usados por diferentes servidores Web.

Los Servlets son un sustituto eficaz de los CGIs : proveen la forma de generar documentos dinámicos que son fáciles de escribir y ejecutar. También evitan el problema de desarrollar la programación según la plataforma utilizada. Los servlets son desarrollados con su propia API, una extensión estándar de Java.

El Contenedor de Servlets (a veces llamado servidor) es el componente encargado de la gestión de los Servlets (instanciar, acceso, destrucción...), y el que controla su ciclo de vida. Dicho contenedor de Servlets, será un programa adicional que hay que instalar para dar soporte

a este tipo de aplicaciones Java en entornos J2EE. Algunos de estos "contenedores" disponibles son MacroMedia JRun, Bea WebLogic, Sun One y uno de los más populares: Apache Tomcat.

Existen una serie de problemas de seguridad y compatibilidad entre Windows Vista y la versión 6.0 de Tomcat. Para solucionar este problema, la mejor opción parece ser instalar un sistema operativo Ubuntu [4] en una maquina virtual dentro la misma maquina física (HOST).

La virtualización se refiere a la abstracción de los recursos de una computadora que crea una capa de la abstracción entre el hardware del host y el sistema operativo de la maquina virtual, siendo un medio para crear una versión virtual de un dispositivo o recurso, como en este caso, un servidor.

El sistema en su conjunto actúa como si realmente existiesen un servidor y un cliente ejecutándose en varias máquinas distintas. Esta arquitectura es fácilmente portable y, en caso de que la aplicación se comercializase, se podría migrar el 'contenedor de servlets' (Tomcat) a un servidor físico independiente.

Para emular este sistema, instalamos el software de Sun Microsystems, VirtualBox que nos permite instalar el sistema operativo Ubuntu, como un "sistema invitado", dentro de otro sistema operativo "anfitrión", que es el Windows Vista.

La sección "Red" de la ventana de configuraciones de la maquina virtual permite ajustar la forma en que VirtualBox presenta las tarjetas de red virtuales a tu VM y como se operan.

- NAT (Network Address Translation) : Es la opción por defecto. De esta forma, el sistema operativo huésped (guest) puede conectarse al mundo exterior usando la red del sistema operativo anfitrión.
- **Interface Anfitrión (Host Interface):** Esta es la opción más recomendable ya que nos permite tener nuestra propia dirección de red (visible desde toda la red). No tiene salida con el exterior, pero establece una red de comunicación entre el propio HOST y la maquina virtual.
- Red Interna (Internal Network): El funcionamiento es muy parecido al Interface Anfitrión (Host Interface) pero con la salvedad que sólo se puede comunicar con otras máquinas virtuales que estén corriendo en el mismo host. Permite mayor velocidad y

fiabilidad, pero tampoco se comunica con el mundo exterior.

La mayor parte de las aplicaciones Web basadas en Servlets se construyen en el marco de trabajo del modelo petición/respuesta HTTP (interacción cliente-servidor).

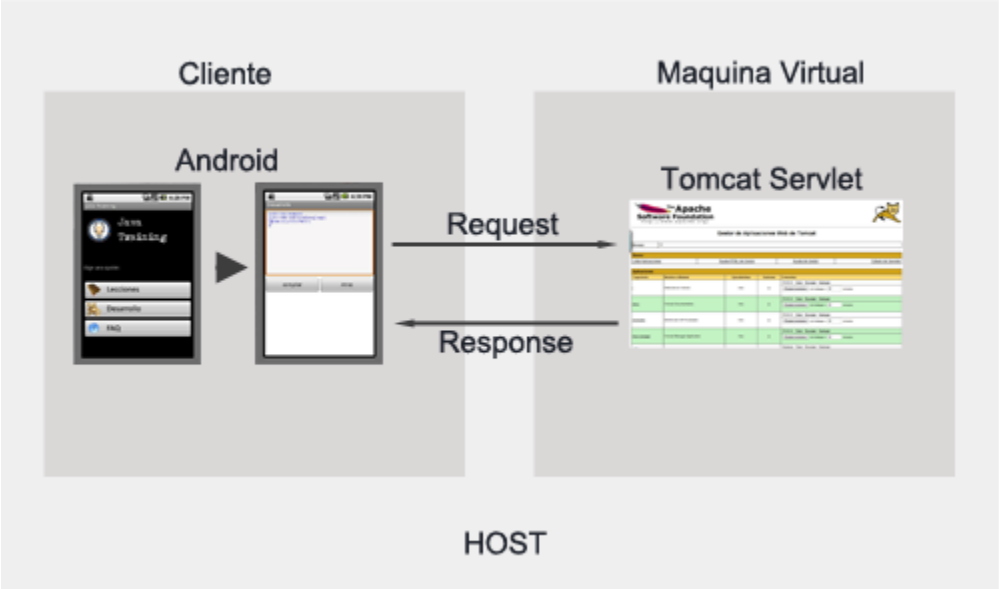



*Ilustración 15: Arquitectura de la aplicación*

### 4.3.1. Cliente

Los servlets pueden obtener fácilmente información acerca del cliente (la permitida por el protocolo HTTP), tal y como su dirección IP, el puerto que se utiliza en la llamada, el método utilizado (GET, POST,...), etc.

El método HTTP POST permite al cliente enviar información al servidor. No tiene la limitación de GET en cuanto a volumen de información transferida, pues ésta no va incluida en el URL de la petición, sino que viaja encapsulada en un 'input stream' que llega al servlet a través de la entrada estándar.

A diferencia de los anteriores métodos, POST no es ni seguro ni idempotente, y por tanto es conveniente su utilización en aquellas aplicaciones que requieran operaciones más complejas que las de sólo-lectura, como por ejemplo modificar bases de datos, etc..

Así pues, definimos este método en el cliente para cuando se pulsa el botón `"compilar"`.

En primer lugar instanciamos el objeto `httpclient` que se ha creado al principio de la aplicación. De esta forma se evita que cada vez que se pulse el botón, suponga un conflicto con cliente ya creado anteriormente. Además cargamos en la variable el texto que se ha introducido en el campo editText.

```
        httpclient = new DefaultHttpClient();
        st1=edt.getText().toString();
```

aquí se crea el objeto `HttpPost` con la dirección del servlet. Si fuese un formulario HTML, esta dirección seria la del parámetro 'action'.

De la misma forma, se definen otros objetos vacíos que corresponden a la respuesta y un objeto Lista para enviar la cadena de texto bajo un identificador.

```
      HttpPost httppost = new
HttpPost("http://192.168.56.102:8080/misServlets/ServletTxt");

        List <NameValuePair> parameters = new ArrayList <NameValuePair>();
        parameters.add(new BasicNameValuePair("st1", st1));

        HttpResponse response = null;
        HttpEntity resEntity = null;
        String res = null;

        try {
        httppost.setEntity(new UrlEncodedFormEntity(parameters));
        response = httpclient.execute(httppost);
        resEntity = response.getEntity();
```

Este objeto `BufferedReader` obtiene la respuesta que se ira leyendo, línea por línea para transcribirla al String `aux`.

```
         BufferedReader b = new BufferedReader(new
InputStreamReader(resEntity.getContent()));
        //Leeríamos la respuesta y haríamos algo con ella, en este caso
únicamente leemos la primera línea
         res = b.readLine();
         String aux="";
        // Mientras se haya leído alguna línea
```

```
while (res!=null)
{
  // Se escribe la línea en pantalla
  aux=aux+res;
  // y se lee la siguiente.
  res = b.readLine();
  }
```

esta cadena de texto, que tiene formato HTML, se 'carga' en el `WebView`, especificando la codificación y el tipo. A continuación se recogen una serie de posibles excepciones.

```
text.loadData(aux, "text/html", "utf-8");

resEntity.consumeContent();
} catch (ClientProtocolException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
```

Cuando la instancia de HTTPClient ya no es necesaria, se cierra el administrador de la conexión para garantizar la inmediata cancelación de asignación de los recursos del sistema

```
httpclient.getConnectionManager().shutdown();
```

### 4.3.2. Servlet

Una vez que se han presentado unas nociones sobre el protocolo HTTP, resulta más sencillo entender las funciones del package javax.servlet.http, que facilitan de sobremanera la creación de servlets que empleen dicho protocolo.

La clase abstracta javax.servlet.http.HttpServlet incluye un número de importante de funciones adicionales e implementa la interface javax.servlet.Servlet. La forma más sencilla de escribir un servlet HTTP es heredando de HttpServlet.

La clase HttpServlet es también una clase abstract, de modo que es necesario definir una clase que derive de ella y redefinir en la clase derivada al menos uno de sus métodos, tales como

doGet(), doPost(), etc.

La clase HttpServlet proporciona una implementación del método service() en la que distingue qué método se ha utilizado en la petición (GET, POST, etc.), llamando seguidamente al método adecuado (doGet() y doPost()).

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletTxt extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
```

Se genera un fichero .java que se rellena con el String enviado con el método `HttpPost` bajo la identificación `'st1'` desde el dispositivo Android

```
        FileWriter fichero = null;
       PrintWriter pwin = null;
       PrintWriter pwout = response.getWriter();

       String codigo = request.getParameter("st1");
       response.setContentType("text/plain");
       try
       {
            fichero = new FileWriter("/tmp/example.java");
            pwin = new PrintWriter(fichero);
            pwin.println(codigo);

       } catch (Exception e) {
           e.printStackTrace();
       } finally {
          try {
           // Nuevamente aprovechamos el finally para
           // asegurarnos que se cierra el fichero.
           if (null != fichero)
            fichero.close();}
           catch (Exception e2) {
           e2.printStackTrace();
           }
    }
```

Una vez cerrado el archivo y guardado en una carpeta temporal de la Maquina Virtual, se crea un `Process` para compilarlo. Para esta operación también se valoró utilizar el método Runtime.exec() pero ya esta en desuso y no es aconsejable.

```
ProcessBuilder pb = new ProcessBuilder("javac","/tmp/example.java");
```

Se obtiene el stream de salida del programa y se pasa a un objeto `BufferedReader` para poder leerlo

```
Process p = pb.start();
 InputStream stream = p.getErrorStream();
InputStreamReader in= new InputStreamReader(stream);
BufferedReader bin= new BufferedReader(in);

String salida="";
String text="";
```

A la hora de transcribir el contenido, se encapsula en unas etiquetas HTML para la posterior lectura en el elemento Android `WebView`

```
    pwout.println(
"<html><body style=\"font-size:10px; color:red;\">Errores:<br/>");
while (true) {
    text=bin.readLine();
            if (text == null) {
        pwout.println("<br/>");
        stream.close();
        break;
    }
            pwout.println(text+"<br/>");
    salida=salida + text ;
}
if (salida==""){
salida="Ninguno";
pwout.println(salida);
}
pwout.println("</body></html>");
pwout.close();
}
```

Con la implementación de este método `doPost()`, nos aseguramos de que la aplicación reconozca los dos tipos de peticiones POST y GET

```
      public void doPost(HttpServletRequest request,HttpServletResponse
response)
        throws ServletException, IOException
    {
        doGet(request, response);
    }
}
```

# 5. Tecnología utilizada

A lo largo del desarrollo del proyecto, se ha tenido que utilizar software que, en ocasiones, no se conocía anteriormente. Este capítulo pretende recoger una introducción a las tecnologías que finalmente se han escogido como soluciones para llevar a cabo la funcionalidad de aplicación.

## 5.1. Eclipse


*Ilustración 16: logo de Eclipse*

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent Azureus (41).

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de

herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

### 5.1.1. Historia de Eclipse

Eclipse comenzó como un proyecto de IBM Canadá. Fue desarrollado por OTI (Object Technology International) como reemplazo de VisualAge también desarrollado por OTI. En noviembre del 2001, se formó un consorcio para el desarrollo futuro de Eclipse como código abierto. En 2003, la fundación independiente de IBM fue creada.

Eclipse 3.0 (2003) seleccionó las especificaciones de la plataforma OSGi [42] como la arquitectura de tiempo de ejecución.

- Callisto

En 2006 la fundación Eclipse coordinó sus 10 proyectos de código abierto, incluyendo la Plataforma 3.2, para que sean liberados el mismo día. Esta liberación simultánea fue conocida como la liberación Callisto.

- Europa

La versión consecutiva a Callisto es Europa, que corresponde a la versión 3.3 de Eclipse, salió el 29 de junio del 2007.

- Ganymede

La versión consecutiva a Europa es Ganymede, que corresponde a la versión 3.4 de Eclipse, salió el 25 de junio del 2008.

- Galileo

La versión consecutiva a Ganymede es Galileo, que corresponde a la versión 3.5 de Eclipse, salió el 24 de junio del 2009.

### 5.1.2. Licencias

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas

licencias son licencias de software libre, pero son incompatibles con Licencia Pública General de GNU (GNU GPL).7 Mike Milinkovich, de la fundación Eclipse comentó que el cambio a la GPL será considerado cuando la versión 3 de la GPL sea liberada.

### 5.1.3. Arquitectura

La base para Eclipse es la Plataforma de cliente enriquecido (del Inglés Rich Client Platform RCP). Los siguientes componentes constituyen la plataforma de cliente enriquecido:

- Plataforma principal - inicio de Eclipse, ejecución de plugins
- OSGi - una plataforma para bundling estándar.
- El Standard Widget Toolkit (SWT)(50) - Un widget toolkit portable.
- JFace - manejo de archivos, manejo de texto, editores de texto
- El Workbench de Eclipse - vistas, editores, perspectivas, asistentes

Los *widgets* de Eclipse están implementados por una herramienta de widget para Java llamada SWT, a diferencia de la mayoría de las aplicaciones Java, que usan las opciones estándar Abstract Window Toolkit (AWT) o Swing. La interfaz de usuario de Eclipse también tiene una capa GUI intermedia llamada JFace, la cual simplifica la construcción de aplicaciones basada en SWT.

El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la plataforma de cliente rico, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente a permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python, permite a Eclipse trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistema de gestión de base de datos. La arquitectura plugin permite escribir cualquier extensión deseada en el ambiente, como sería Gestión de la configuración. Se provee soporte para Java y CVS en el SDK de Eclipse. Y no tiene por qué ser usado únicamente para soportar otros lenguajes de programación.

La definición que da el proyecto Eclipse acerca de su software es: "una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular".

En cuanto a las aplicaciones clientes, eclipse provee al programador con *frameworks* muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones Web, etc. Por ejemplo, GEF (Graphic Editing Framework - Framework para la edición gráfica) es un plugin de Eclipse para el desarrollo de editores visuales que pueden ir desde procesadores de texto wysiwyg hasta editores de diagramas UML, interfaces gráficas para el usuario (GUI), etc. Dado que los editores realizados con GEF "viven" dentro de Eclipse, además de poder ser usados conjuntamente con otros plugins, hacen uso de su interfaz gráfica personalizable y profesional.

El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. El IDE también hace uso de un espacio de trabajo, en este caso un grupo de metadata en un espacio para archivos plano, permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente.

El 28 de junio de 2005 fue liberada la versión 3.1 que incluye mejoras en el rendimiento, el soporte de Java 5.0, mejor integración con Ant (incluido debugger) y CVS.

### 5.1.3. Características

Eclipse dispone de un editor de texto con resaltado de sintaxis. La compilación es en tiempo real. Tiene pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes (wizards) para creación de proyectos, clases, tests, etc., y refactorización.

Además, están disponibles una serie de "plugins" con los que es posible añadir un control de versiones con Subversion, por ejemplo, e integración con Hibernate.

## 5.2. VirtualBox


*Ilustración 17: logo de VirtualBox*

Sun xVM VirtualBox es un software de virtualización para arquitecturas x86(47) que fue desarrollado originalmente por la empresa alemana Innotek GmbH, pero que pasó a ser propiedad de la empresa Sun Microsystems en febrero de 2008 cuando ésta compró a innotek. Por medio de esta aplicación es posible instalar sistemas operativos adicionales, conocidos como "sistemas invitados", dentro de otro sistema operativo "anfitrión", cada uno con su propio ambiente virtual. Por ejemplo, se podrían instalar diferentes distribuciones de GNU/Linux en VirtualBox instalado en Windows XP o viceversa.

Entre los sistemas operativos soportados (en modo anfitrión) se encuentran GNU/Linux, Mac OS X, OS/2 Warp , Windows, ySolaris/OpenSolaris, y dentro de éstos es posible virtualizar los sistemas operativos FreeBSD, GNU/Linux, OpenBSD, OS/2 Warp, Windows, Solaris, MS-DOS y muchos otros.

### 5.2.1. Historia

La aplicación fue inicialmente ofrecida bajo una licencia de software privado, pero en enero de 2007, después de años de desarrollo, surgió VirtualBox OSE (Open Source Edition) bajo la licencia GPL 2. Actualmente existe la versión privada, VirtualBox, que es gratuita únicamente bajo uso personal o de evaluación, y esta sujeta a la licencia de "Uso Personal y de

Evaluación VirtualBox" (VirtualBox Personal Use and Evaluation License o PUEL) y la versión Open Source, VirtualBox OSE, que es software libre, sujeta a la licencia GPL.

### 5.2.2. Características

Algunas de las características de VirtualBox son:

- **Modularidad**. VirtualBox tiene un diseño extremadamente modular con interfaces bien definidas de programación interna y un diseño cliente / servidor. Esto hace que sea fácil de controlar desde varias interfaces a la vez: por ejemplo, puede iniciar una máquina virtual en una máquina virtual típica interfaz gráfica de usuario ya continuación, el control de la máquina de la línea de comandos, o posiblemente de forma remota. VirtualBox también viene con un completo kit de desarrollo de software: a pesar de que es Open Source Software, no se tiene que 'hackear' la fuente para escribir una nueva interfaz.
- **Descripciones de máquinas virtuales en XML.** Las opciones de configuración de las máquinas virtuales se almacenan enteramente en XML y son independientes de las máquinas locales. Las definiciones de máquina virtual pueden ser fácilmente portadas a otros equipos.
- **Adiciones de huéspedes** (Guest Additions) para Windows, Linux y Solaris. VirtualBox tiene software especial que puede ser instalado dentro de Windows, Linux y Solaris virtuales para mejorar el rendimiento y conseguir una integración mucho más transparente. Entre las características proporcionadas por estas adiciones de los huéspedes están la integración puntero del ratón y las soluciones arbitrarias de pantalla (por ejemplo, cambiar el tamaño de la ventana de resultados). Hay también ampliaciones para OS / 2 pero con la funcionalidad algo reducida.
- **Carpetas compartidas**. Al igual que muchas otras soluciones de virtualización, para el intercambio fácil de datos entre anfitriones y huéspedes, VirtualBox permite declarar ciertos directorios de acogida como "carpetas compartidas", que luego se pueden acceder desde dentro de las máquinas virtuales.
  Una serie de características adicionales disponibles con el lanzamiento de la versión

completa de VirtualBox:

- **Virtual USB(45) controles.** VirtualBox implementa un controlador virtual USB y le permite conectar dispositivos USB a sus máquinas virtuales sin tener que instalar controladores de dispositivos específicos en el host.
- **Protocolo de escritorio remoto.** A diferencia de cualquier otro software de virtualización, VirtualBox es plenamente compatible con el estándar de Remote Desktop Protocol (RDP) (46). Una máquina virtual puede actuar como un servidor RDP, lo que le permite "correr" la máquina virtual de forma remota en un cliente ligero que sólo muestra los datos de RDP.
- **USB sobre RDP.** Con esta característica única, una máquina virtual que actúa como un servidor RDP puede tener acceso arbitrario los dispositivos USB que están conectados en el cliente RDP. De esta manera, una máquina potente servidor pueden virtualizar una gran cantidad de clientes ligeros que sólo necesitan mostrar los datos RDP y que sus dispositivos USB están conectados.

### 5.2.3. Arquitectura

En cuanto a la emulación de hardware, los discos duros de los sistemas invitados son almacenados en los sistemas anfitriones como archivos individuales en un contenedor llamado Virtual Disk Image, incompatible con los demás software de virtualización.

Otra de las funciones que presenta es la de montar imágenes ISO(48) como unidades virtuales de CD o DVD, o como un disco floppy.

## 5.3. Tomcat

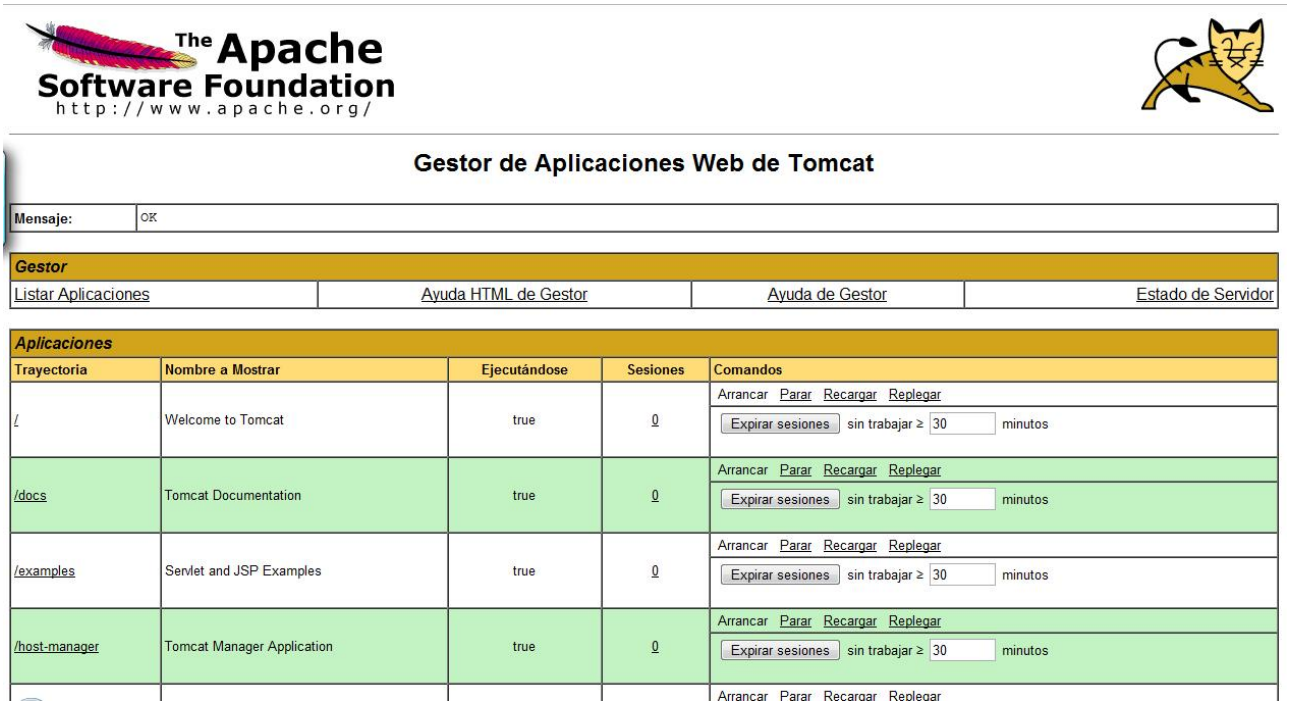



*Ilustración 18: los Servlets desde un navegador en la sesión de Administrador*

Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

Tomcat es un servidor Web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor Web Apache.

Tomcat puede funcionar como servidor Web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor Web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que

disponga de la máquina virtual Java.

### 5.3.1. Historia de Apache Tomcat

Tomcat empezó siendo una implementación de la especificación de los servlets comenzada por James Duncan Davidson, que trabajaba como arquitecto de software en Sun Microsystems y que posteriormente ayudó a hacer el proyecto open source y en su donación a la Apache Software Foundation.

Duncan Davidson inicialmente esperaba que el proyecto se convirtiese en open source y dado que la mayoría de los proyectos open source tienen libros de O'Reilly asociados con un animal en la portada, quiso ponerle al proyecto nombre de animal. Eligió Tomcat (gato), pretendiendo representar la capacidad de cuidarse por sí mismo, de ser independiente.

### 5.3.2. Estado de su desarrollo

Tomcat es mantenido y desarrollado por miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software Licence. Las primeras distribuciones de Tomcat fueron las versiones 3.0.x. Las versiones más recientes son las 6.x, que implementan las especificaciones de Servlet 2.5 y de JSP 2.1. A partir de la versión 4.0, Jakarta Tomcat utiliza el contenedor de servlets Catalina.

### 5.3.3. Estructura de directorios

La jerarquía de directorios de instalación de Tomcat incluye:

- bin - arranque, cierre, y otros scripts y ejecutables
- common - clases comunes que pueden utilizar Catalina y las aplicaciones Web
- conf - ficheros XML y los correspondientes DTD(43) para la configuración de Tomcat
- logs - logs de Catalina y de las aplicaciones
- server - clases utilizadas solamente por Catalina
- shared - clases compartidas por todas las aplicaciones Web

- webapps - directorio que contiene las aplicaciones Web
- work - almacenamiento temporal de ficheros y directorios

### 5.3.4. Características

Tomcat 3.x (distribución inicial)

- Implementado a partir de las especificaciones Servlet 2.2 y JSP 1.1(44)
- Recarga de servlets
- Funciones básicas HTTP

Tomcat 4.x

- Implementado a partir de las especificaciones Servlet 2.3 y JSP 1.2
- Contenedor de servlets rediseñado como Catalina
- Motor JSP rediseñado con Jasper
- Conector Coyote
- Java Management Extensions (JMX), JSP Y administración basada en Struts

Tomcat 5.x

- Implementado a partir de las especificaciones Servlet 2.4 y JSP 2.0
- Recolección de basura reducida
- Capa envolvente nativa para Windows y Unix para la integración de las plataformas
- Análisis rápido JSP

**Tomcat 6.x**

- Implementado de Servlet 2.5 y JSP 2.1
- Soporte para Unified Expression Language 2.1
- Diseñado para funcionar en Java SE 5.0 y posteriores
- Soporte para Comet a través de la interfaz CometProcessor

## 5.4. Servlets

Los servlets son módulos de código escrito en Java que añaden funcionalidad a un servidor Web. Fueron diseñados para aceptar peticiones de un cliente y generar los mensajes de respuesta correspondiente.

Los servlets no especifican ningún tipo de protocolo entre el cliente y el servidor, únicamente se limitan a recoger peticiones de usuario de una página HTML y generar respuestas.

### 5.4.1. Características

Entre las características principales de los servlets cabe citar las siguientes:

- Son independientes del servidor utilizado y de su sistema operativo, lo que quiere decir que a pesar de estar escritos en Java, el servidor puede estar escrito en cualquier lenguaje de programación.
- Los servlets pueden llamar a otros servlets, e incluso a métodos concretos de otros servlets (en la misma máquina o en una máquina remota). De esta forma se puede distribuir de forma más eficiente el trabajo a realizar. Por ejemplo, se podría tener un servlet encargado de la interacción con los clientes y que llamara a otro servlet para que a su vez se encargara de la comunicación con una base de datos.
- Los servlets pueden obtener fácilmente información acerca del cliente (la permitida por el protocolo HTTP), tal como su dirección IP, el puerto que se utiliza en la llamada, el método utilizado (GET, POST), etc.
- Permiten además la utilización de cookies y sesiones, de forma que se puede guardar información específica acerca de un usuario determinado, personalizando de esta forma la interacción cliente/servidor. Una clara aplicación es mantener la sesión con un cliente.
- Los servlets pueden actuar como enlace entre el cliente y una o varias bases de datos en arquitecturas cliente-servidor.
- Asimismo, pueden realizar tareas de proxy para un applet. Debido a las restricciones de seguridad, un applet no puede acceder directamente por ejemplo a un servidor de datos

localizado en cualquier máquina remota, pero sí podría hacerlo a través de un servlet.

- Permiten la generación dinámica de código HTML, lo que se puede utilizar para la creación de contadores, banners, etc.

### 5.4.2. Funcionamiento

Como se ha mencionado en los puntos anteriores un servlet añade funcionalidad a una página Web. No siempre realizan las mismas tareas, es decir, no tienen un patrón fijo de trabajo, pero de forma general las podemos dividir de la siguiente forma:

1. Leer cualquier dato enviado por el usuario: los datos normalmente se introducen por medio de la página Web, pero también pueden obtenerse a partir de un applet Java.
2. Obtener otra información sobre la petición que se encuentra embebida en la propia petición HTTP: esta información se refiere por ejemplo a los cookies, el nombre del host de donde proviene la petición, etc.
3. Generar los resultados: esta parte puede requerir acceder a una base de datos, ejecutar una llamada RMI o CORBA, invocar a una aplicación o simplemente computar los datos de entrada.
4. Generar un documento con los resultados: debemos establecer el tipo de documento que va a ser devuelto (una página HTML, una imagen, un archivo comprimido, etc.).
5. Establecer los parámetros apropiados para la respuesta.
6. Enviar la respuesta al cliente: una vez que tenemos el formato del documento que entregaremos como respuesta y tenemos establecidos los parámetros de la comunicación enviamos la respuesta al cliente.

En la siguiente figura se muestra de forma genérica cómo funciona un servlet:

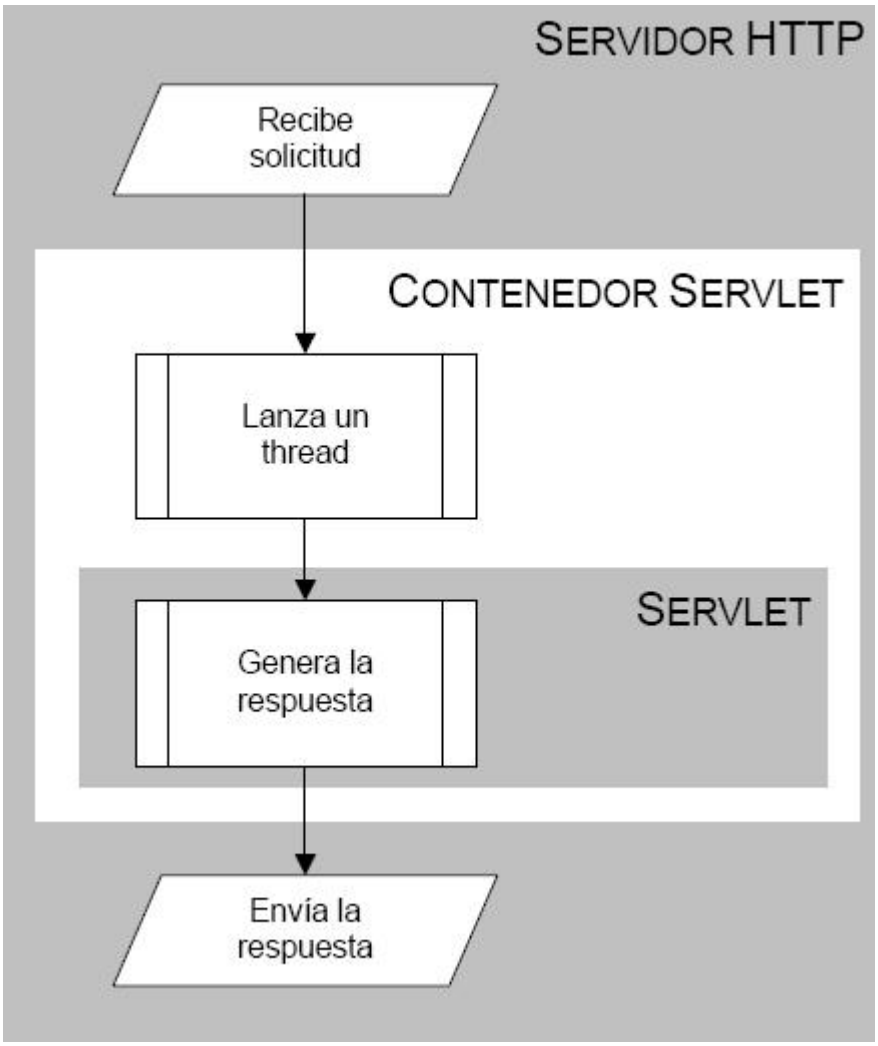



*Ilustración 19: esquema simple del funcionamiento de un Servlet*

### 5.4.3. Servlets vs. CGI

Los primeros servidores HTTP no incluían ningún mecanismo para generar respuestas dinámicamente, por lo tanto se desarrollaron interfaces para comunicar el servidor con programas externos que implementan dicha funcionalidad.

La especificación CGI describe una interfaz estándar que sirve para que un servidor Web envíe solicitudes del navegador al programa CGI, y para que el programa CGI devuelva datos de respuesta al navegador a través del servidor Web.

Los servlets ofrecen la misma funcionalidad que los CGI pero evitan ciertos problemas que tenían éstos últimos. Entre las ventajas que ofrecen los servlets cabe destacar:

- Con CGI cada vez que se realizaba una petición se creaba un nuevo proceso para atenderla, con las consecuencias que conlleva (tiempo para el cambio de proceso, memoria ocupada, etc.). Esto no ocurre con los servlets, pues con cada petición no se genera un nuevo proceso, sino un nuevo hilo de ejecución.
- Si tenemos varias peticiones simultáneas y estamos utilizando CGI tendremos en memoria tantas instancias del programa como peticiones, lo que conlleva un gasto de memoria enorme si este número es muy grande y una degradación continua del rendimiento. Con los servlets, al tener hilos de ejecución, con una sola instancia de nuestro programa nos basta, ahorrando así muchos recursos.
- Cuando un programa CGI termina de responder a una petición, el programa finaliza. Esto hace que cada vez que se realiza una nueva petición todo el programa tiene que ser cargado de nuevo, establecer de nuevo las conexiones con las bases de datos, etc. Los servlets, sin embargo, permanecen en memoria entre peticiones, lo que hace que se gane mucho tiempo en las sucesivas peticiones porque el programa ya está cargado en memoria.
- Los servlets favorecen la independencia de la plataforma, ya que están escritos en Java y por tanto siguen el estándar API. Como consecuencia los servlets están escritos para ser ejecutados en una cantidad enorme de servidores (Apache, Microsoft IIS, Java Web Server, etc.).
- Un servlet puede ejecutarse en una sandbox [49] o recinto de seguridad parecido al modelo que se sigue con los applets. Debido a esto pueden colocarse servlets en servidores dedicados a hosting sin que la empresa tema por la integridad del servidor y la seguridad de las aplicaciones.
- Una vez que tienes un servidor Web añadir un servlet supone un ligero incremento en el coste, a diferencia de los programas CGI, que necesitan una gran inversión para obtenerlos.

### 5.4.4. Paquetes Java para servlets

Existen una serie de paquetes escritos en Java que permiten trabajar de forma fácil y

cómoda con los servlets: *javax.servlet y javax.servlet.http*.

El paquete j**avax.servlet** define las siguientes clases e interfaces:

- Clases:
  - GenericServlet: implementa la interfaz Servlet.
  - ServletInputStream: se utiliza para acceder a información de las solicitudes Web.
  - ServletOutputStream: se utiliza para enviar la respuesta a un cliente.
- Interfaces:
  - Servlet: debe ser implementada por todos los servlets. Se utiliza para iniciar y detener el servlet.
  - ServletRequest: encapsula la solicitud de servicio de un cliente.
  - ServletResponse: es utilizada por el servlet para responder a una solicitud.
  - ServletConfig: para que el servlet conozca información del contenedor servlet.
  - ServletContext: define el entorno en el que se ejecuta un servlet.
  - SingleThreadModel: se usa para identificar a los servlets que no pueden ser accedidos por más de un hilo al mismo tiempo.

El paquete **javax.servlet.http** se usa para definir servlets específicos de HTTP y define las siguientes clases e interfaces:

- Clases:
  - Cookie: representa una cookie HTTP.
  - HttpServlet: amplía la clase GenericServlet con el fin de utilizar las interfaces HttpServletRequest y HttpServletResponse.
  - HttpSessionBindingEvent: implementa el evento que se genera cuando un objeto está ligado o se desliga de una sesión HTTP.
  - HttpUtils: analizar sintácticamente una cadena de consulta.

- Interfaces:
    - HttpServletRequest: amplía la interfaz ServletRequest agregando métodos para acceder a detalles de una solicitud HTTP.
    - HttpServletResponse: amplía la interfaz ServletResponse para devolver respuestas específicas de HTTP.
    - HttpSession: implementa los servlets que permiten dar soporte a las sesiones entre el navegador y el servidor.
    - HttpSessionBindingListener: la implementan clases cuyos objetos están asociados a sesiones HTTP.
    - HttpSessionContext: se utiliza para representar a una colección de objetos que están asociados a los identificadores de sesión.

# 6. Planificación temporal

## 6.1. Desglose por fases del proyecto

En la tabla 6.1 se especifican las actividades que se han llevado a cabo para la realización del proyecto, incluyendo las horas que han sido necesarias para llevar a desarrollar cada una de ellas.

El tiempo empleado total han sido 680 horas aproximadamente.

| Fases | Horas |
|---|---|
| Lectura de documentación Android | 40 horas |
| Instalación de las herramientas de desarrollo | 16 horas |
| Familiarización con la plataforma de desarrollo | 40 horas |
| Recopilación de material docente Java | 24 horas |
| Análisis y diseño de la aplicación | 80 horas |
| Implementación | 360 horas |
| Pruebas | 40 horas |
| Documentación | 80 horas |

*Tabla 1: Fases del proyecto*

## 6.2. Diagrama de GANTT

El diagrama de Gantt sitúa de forma secuencia las diferentes etapas del proyecto a lo largo de una línea temporal que representa el tiempo total del proyecto.

| Nombre | Duración | Inicio |
|---|---|---|
| Lectura de documentación... | 5 days | 17/08/09 8:00 |
| *Analista* | *5 days* | *17/08/09 8:00* |
| *Programador* | *5 days* | *17/08/09 8:00* |
| Instalación de las herramie... | 2 days | 22/08/09 7:00 |
| *Programador* | *2 days* | *22/08/09 7:00* |
| Familiarización con la plata. | 5 days | 26/08/09 7:00 |
| *Programador* | *5 days* | *26/08/09 7:00* |
| Recopilación de material d... | 3 days | 2/09/09 8:00 |
| *Documentacion* | *3 days* | *2/09/09 8:00* |
| Analisis y diseño de la aplic... | 10 days | 7/09/09 7:00 |
| *Analista* | *10 days* | *7/09/09 7:00* |
| Implementación | 45 days | 21/09/09 7:00 |
| *Programador* | *45 days* | *21/09/09 7:00* |
| Pruebas | 2,5 days | 23/11/09 8:00 |
| *Analista* | *2,5 days* | *23/11/09 8:00* |
| *Programador* | *2,5 days* | *23/11/09 8:00* |
| Documentación | 10 days | 30/11/09 8:00 |
| *Documentacion* | *10 days* | *30/11/09 8:00* |

# 7. Presupuesto

En el siguiente apartado se va a mostrar el presupuesto requerido para el proyecto, desglosando los costes del desarrollo de cada tarea.

## 7.1. Gastos de persona imputable al proyecto

| Cargo | Horas (h) | Costo (€) |
|---|---|---|
| **Analista** | **110** | **3.960 €** |
| • Lectura de documentación de Android | 20 | 720 **€** |
| • Análisis y diseño de la aplicación | 80 | 2.880 **€** |
| • Pruebas | 10 | 360 **€** |
| **Programador** | **446** | **13.380 €** |
| • Implementación | 360 | 10.800 **€** |
| • Familiarización con la plataforma de desarrollo | 40 | 1200 **€** |
| • Instalación de las herramientas de desarrollo | 16 | 480 **€** |
| • Pruebas | 10 | 300 **€** |
| • Lectura de documentación de Android | 20 | 600 **€** |
| **Documentación** | **104** | **2.704 €** |
| • Documentación | 80 | 2.080 **€** |
| • Recopilación de material docente de Java | 24 | 624 **€** |
| **Total** | **660 horas** | **20044 €** |

*Tabla 2: Gastos por personal*

## 7.2. Recursos materiales empleados

En la tabla siguiente se detallan los costes de los recursos necesarios para elaboración de este proyecto.

| Recurso | Unidades | Coste (€ ) |
|---|---|---|
| Portátil Intel Core Duo | 1 | 800 **€** |
| Licencia Windows Vista | 1 | 200 **€** |
| **Total** | | **1.000 €** |

*Tabla 3: Gastos por recursos materiales*

El resto del software que se ha utilizado es freeware, software libre. Desde el entorno de desarrollo Eclipse, el contenedor de servlets Tomcat, la maquina virtual VirtualBox y el sistema operativo instalado en ella (Ubuntu), así como el Editor de texto OperOffice para redactar este documento y el programa OpenProj para realizar la planificación temporal de recursos y el presupuesto.

En caso de migrar esta maqueta a un escenario real, habría que contar con más elementos a incluir en el presupuesto como, por ejemplo, un servidor, un móvil Android, cables de red..

## 7.3. Resumen y beneficio

La tabla siguiente resume todos los costes que ha requerido el proyecto, así como la suma total de los mismos

| Recurso | Coste |
|---|---|
| Gastos de personal | 20.044,00 € |
| Recursos materiales | 1.000,00 € |
| **Total** | **21.044,00 €** |

*Tabla 4: Resumen del presupuesto*

Finalmente se calculan y se añaden los beneficios a obtener con el proyecto, un 12% sobre la suma del coste total. Sobre esta cantidad bruta, se incluye el Impuesto sobre el Valor Añadido, el 16%.

| Costes | Coste |
|---|---|
| Coste de Recursos | 21.044 € |
| Beneficio | 2.525,28 € |
| IVA | 3.771,1 € |
| **Total** | **27340,4 €** |

*Tabla 5: Coste total*

# 8. Conclusiones y trabajo futuro

En la etapa final de este documento, una vez desarrollado todo el trabajo, se lleva a cabo un análisis de los resultados para evaluar la aplicación y definir posibles modificaciones futuras.

## 8.1. Conclusiones

En primer lugar, se establece una comparativa con los objetivos marcados al inicio del proyecto. Básicamente se pueden dividir en dos grupos: por un lado se trataba de afianzar conocimientos previos, conocer nuevas tecnologías e investigar y recoger documentación sobre el desarrollo en los entornos de trabajo que se han utilizado. En este aspecto, estamos bastante satisfechos tras la 'experiencia Android'. Entendemos que se trata de una plataforma en pleno crecimiento y este primer contacto puede ser muy útil como experiencia laboral. De la misma forma, el hecho de haber tenido que estudiar distintos software para buscar soluciones no ha hecho mas que incrementar la capacidad de trabajo, así como agilidad para estudiar los recursos disponibles.

Por otro lado, el otro objetivo principal es crear la aplicación en si. En este aspecto, se ha conseguido la maqueta de una aplicación funcional y real con muchas posibilidades en el campo de la docencia. Esperamos que en un futuro, tanto el manual como la aplicación sean útiles como elemento de consulta y base de desarrollo para futuras aplicaciones.

El desarrollo del trabajo ha sido bastante ajustado al planteamiento inicial del proyecto aunque en ocasiones se han tenido que estudiar otras soluciones ante problemas imprevistos:

La SDK de Android no es todo lo estable que debería ser. Las continuas actualizaciones y los problemas al migrar la aplicación de una a otra, hicieron que algunas funcionalidades que ya estaban terminadas tuviesen que ser revisadas.

En todo momento se ha tenido como prioridad buscar soluciones que potenciasen la capacidad de portabilidad y compatibilidad tanto de la aplicación en el servidor, como de la que está en el dispositivo cliente. Esta es una de las razones por las que se eligió implementar un servlet ya que garantiza la portabilidad a distintos tipos de plataforma. Pero el contenedor de servlets, Tomcat 6.0 presentó una serie de problemas de seguridad y compatibilidad al intentar

instalarlo en Windows Vista. Por esta razón, se optó por utilizar una maquina virtual con Ubuntu que, a priori, no presentaba ninguna problema ya que esta basado en Linux.

## 8.2. Trabajo futuro

Como ya se ha comentado a lo largo de este documento, esta aplicación es una maqueta con la finalidad de que se implemente en un escenario real. Tal y como se presenta en este proyecto, ya es susceptible de subirla a la plataforma Android Market para su distribución. Y, por otro lado, la parte del Servlet, migrarla a un servidor.

Bien es cierto, que existen una serie de ampliaciones y posibilidades que no están en la versión final de la aplicación pero que, en un futuro, seria interesante estudiarlas:

- Acceder a la aplicación a través de un *log-in*. Es decir, utilizar un usuario y contraseña y mantener la sesión. Esta funcionalidad es interesante para recabar información sobre el usuario: almacenar archivos '.java', programar juegos de habilidad con puntuación y rankings, establecer un seguimiento del usuario etc.. Es factible ya que la clase `httpSession` de `httpServlets` tiene esta capacidad (como ya queda explicado en el punto 5.4.4. Clases Java para Servlets)
- Como hemos comentado en el punto anterior, la aplicación esta abierta a posibles extensiones y ampliaciones. Por ahora, *Java Training* se limita a una colección de temario y a la funcionalidad de desarrollo, pero sería interesante implementar una funcionalidad de 'juegos de habilidad' para practicar los ejercicios de forma mas entretenida, por ejemplo.
- El apartado de Desarrollo compila un archivo Java y devuelve los errores (si los hay). Otra posibilidad a estudiar es que compile y ejecute este código mostrándolo por pantalla. Esto no es trivial, ya que Android no trabaja directamente con una Maquina Virtual de Java, sino con una versión más ligera y limitada que se llama Dalvik.

# 9. Anexos

## Anexo I: Manual para el usuario

Hemos llegado a un punto en este documento que ya se conoce como instalar la aplicación, así como todas las tecnologías que la soportan y ponerlo en marcha. Ahora vamos a explorar la aplicación a nivel de usuario.

El punto de partida es el emulador de Android. Gráficamente reproduce una pantalla de móvil con el escritorio, las aplicaciones, menús, etc.. En la parte derecha hay un teclado con unos controles genéricos de *smartphone* y debajo un teclado alfanumérico.


*Ilustración 20: Vista completa del emulador*

Al pulsar el menú, se despliega el catálogo de aplicaciones, y ahí se encuentra un icono con el escudo de la Universidad Carlos III de Madrid sobre el titulo *Java Training*. Esa es la aplicación.

Entramos en la aplicación y llegamos a una pantalla de bienvenida que a su vez es el menú (esto queda explicado en el punto 4.2.1 menú de inicio). En la parte superior está el logotipo de la aplicación y en la mitad inferior se encuentran los tres botones del menú.


*Ilustración 21: Vista del portada de bienvenida*

**Lecciones**

En este apartado, al acceder, nos encontramos con una colección de temario de consulta para aprender Java. El modelo de organización jerárquico ya ha quedado explicado en el punto 4.2.2.

Se trata de una lista seleccionable, así que consiste en acceder al tema que interese. Esta aplicación es una maqueta, pero en una versión real, se podría incluir una pequeña descripción que acompañe al titulo para que resulte más claro.


*Ilustración 22: Vista del menú lecciones*

Una vez dentro de de la lección deseada, nos encontramos con un visor a pantalla completa y una pequeña galería horizontal en la parte inferior. Se trata de mostrar a tamaño grande la página seleccionada, mientras las demás esperan ordenadas.

El móvil Android tiene capacidad táctil, así que se ha aprovechado esta característica para realizar un efecto deslizante en la galería inferior. De esta forma, se agiliza mucho el visionado o búsqueda de una pagina completa, sin tener que abrir cada pagina a modo de 'pase de diapositivas'.


*Ilustración 23: Visor de lecciones*

### Desarrollo

En esta funcionalidad, se accede directamente a una consola para compilar código Java. En la mitad superior esta el campo de texto para introducir el código Java.


*Ilustración 24: Vista de la consola de desarrollar antes de compilar*

A continuación, una vez pulsado el botón de compilar, se establece una conexión con el Servlet, enviándole nuestro código como una cadena de texto (String). Durante estos segundos, la aplicación espera.

La respuesta se recibe en un elemento Web, que mantiene la forma de visualizarlo que obtendríamos si compilásemos desde una consola Linux.

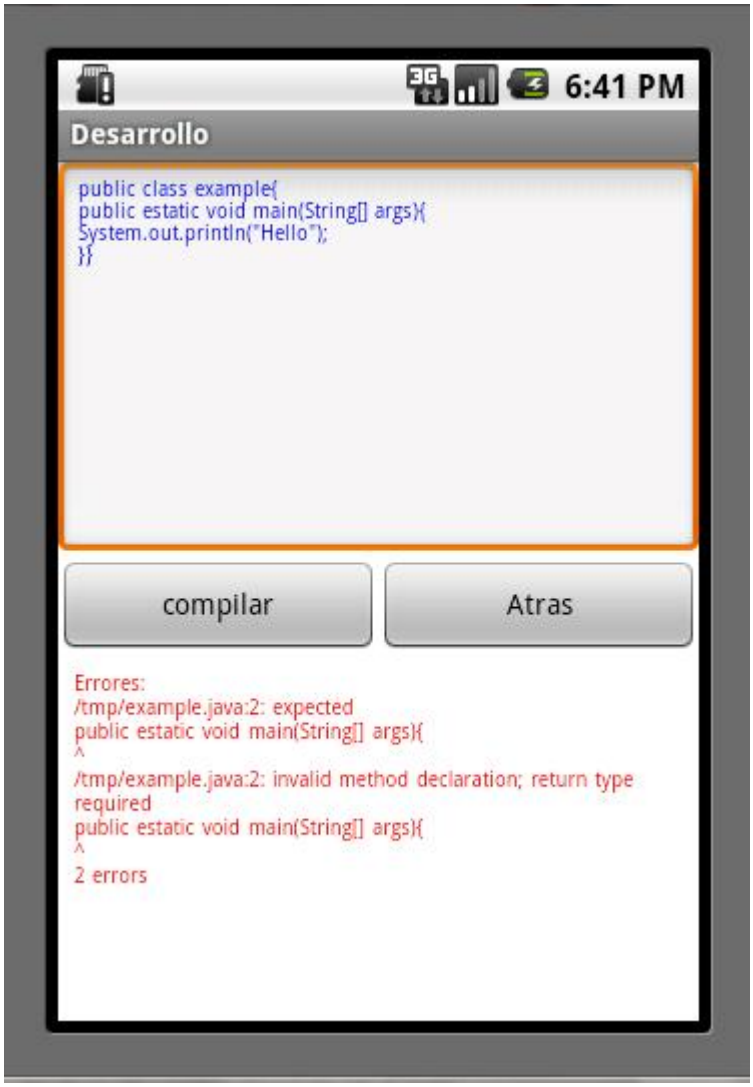



*Ilustración 25: Vista de la consola de desarrollar con errores*

Si se ha programado el archivo java correctamente, el compilador nos devuelve un mensaje.


*Ilustración 26: Vista de la consola de desarrollar sin errores*

La aplicación es completamente navegable en las dos direcciones. En algunos casos, las pantallas están reforzadas con un botón ATRÁS, aunque en toda la aplicación el botón KeyBack del teclado está implementado. Por defecto, se salía de la aplicación al pulsarlo.

## Anexo II: Glosario de términos

[1] **iPhone:** es un teléfono inteligente multimedia con conexión a internet, pantalla táctil con tecnología multitáctil de la compañía Apple Inc.

[2] **SDK:** (el software development kit) es generalmente un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, *frameworks*, plataformas de hardware, ordenadores, videoconsolas, sistemas operativos, etc.

[3] **NDK:** compañero natural del SDK, proporciona las herramientas necesarias para generar e incrustar código máquina ARM (Advanced RISC Machines, familia de microprocesadores RISC) nativo en las aplicaciones.

[4] **Ubuntu:** un sistema operativo actualizado y estable basado el Linux

[5] **kernel:** es la parte fundamental de un sistema operativo. Es el software responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma más básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema.

[6] **Smartphone:** (teléfono inteligente en español) es un dispositivo electrónico que funciona como un teléfono móvil con características similares a las de un ordenador personal.

[7] **Symbian:** es un sistema operativo que fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran Nokia, Sony Ericsson, PSION, Samsung, etc..

[8] **Blackberry:** es una línea de dispositivos handheld inalámbricos introducida en 1999. Estos dispositivos entre otras funciones admiten correo electrónico, telefonía móvil, SMS, navegación Web y otros servicios de información inalámbricos.

[9] **Windows mobile:** es un sistema operativo compacto, con una suite de aplicaciones básicas para dispositivos móviles basados en la API Win32 de Microsoft.

[10] **C/C++:** lenguajes de programación orientados a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se

utiliza para crear aplicaciones.

[11] **Hal de Linux:** La capa de abstracción de hardware o HAL (acrónimo del inglés de hardware abstraction layer) es un elemento del sistema operativo que funciona como una interfaz entre el software y el hardware del sistema, proveyendo una plataforma de hardware consistente sobre la cual correr las aplicaciones.

[12] **MPEG4:** introducido a finales de 1998, es el nombre de un grupo de estándares de codificación de audio y video así como su tecnología relacionada normalizada por el grupo MPEG(Moving Picture Experts Group) de ISO/IEC. Los usos principales del estándar MPEG-4 son los flujos de medios audiovisuales, la distribución en CD, la transmisión bidireccional por videófono y emisión de televisión.

[13] **H.264:** es una norma que define un códec de vídeo de alta compresión, desarrollada conjuntamente por el ITU-T Video Coding Experts Group (VCEG) y el ISO/IECMoving Picture Experts Group (MPEG). La intención del proyecto H.264/AVC fue la de crear un estándar capaz de proporcionar una buena calidad de imagen con tasas binarias notablemente inferiores a los estándares previos

[14] **MP3:** es un formato de audio digital comprimido con pérdida

[15] **OGG:** es un formato libre de archivo contenedor multimedia, diseñado para dar un alto grado de eficiencia en el "streaming" y la compresión de archivos.

[16] **AAC:** es un formato informático de señal digital audio basado en el Algoritmo de compresión con pérdida, un proceso por el que se eliminan algunos de los datos de audio para poder obtener el mayor grado de compresión posible, resultando en un archivo de salida que suena lo más parecido posible al original.

[17] **AMR:** Multi-tasa adaptativo (en inglés Adaptive Multi-Rate, AMR) es un formato de compresión de audio optimizado para la codificación de voz.

[18] **JPG:** es un algoritmo de compresión con pérdida y además a menudo es considerado también un formato de archivo

[19] **PNG:** es un formato gráfico basado en un algoritmo de compresión sin pérdida para bitmaps no sujeto a patentes

[20] **GIF:** es un formato gráfico utilizado ampliamente en la World Wide Web, tanto para imágenes como paraanimaciones.

[21] **GSM:** El Sistema Global para las Comunicaciones Móviles (GSM, proviene de "Groupe Special Mobile") es un sistema estándar, completamente definido, para la comunicación mediante teléfonos móviles que incorporan tecnología digital.

[22] **Bluetooth:** es una especificación industrial para Redes Inalámbricas de Área Personal (WPANs) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz.

[23] **EDGE:** es una tecnología de la telefonía móvil celular que se considera una evolución del GPRS (General Packet Radio Service). Esta tecnología funciona con redes GSM.

[24] **3G:** es la abreviación de tercera-generación en telefonía móvil. Los servicios asociados con la tercera generación proporcionan la posibilidad de transferir tanto voz y datos (una llamada telefónica) y datos no-voz (como la descarga de programas, intercambio de email, y mensajería instantánea).

[25] **WiFi:** es un sistema de envío de datos sobre redes computacionales que utiliza ondas de radio en lugar de cables

[26] **GPS:** es un sistema global de navegación por satélite (GNSS) que permite determinar en todo el mundo la posición de un objeto, una persona, un vehículo o una nave

[27] **Stack:** es una lista ordinal o estructura de datos en la que el modo de acceso a sus elementos es de tipo LIFO (del inglés last in first out, es decir, "último en entrar, primero en salir") que permite almacenar y recuperar datos

[28] **bytecode:** es un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario producido por el compilador cuyo contenido es el código objeto o código máquina

[29] **CDMA /EVDO**: es un estándar de telecomunicaciones para la transmisión inalámbrica de datos a través de redes de telefonía celular evolucionadas que utiliza técnicas de multiplexaciono control de acceso al medio basados en la tecnología de espectro expandido.

[30] **802.1x:** La IEEE 802.1X es una norma del IEEE para el control de acceso a red basada en puertos.

[31] **VPN:** La Red Privada Virtual (RPV), en inglés Virtual Private Network (VPN), es una tecnología de red que permite una extensión de la red local sobre una red pública o no controlada, como por ejemplo Internet.

[32] **IDE:** es un programa informático compuesto por un conjunto de herramientas de programación. Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, al que mediante plugins se le puede añadir soporte de lenguajes adicionales.

[33] **ADT:** plugin con las herramientas de Android para el software de desarrollo Eclipse

[34] **XML**: es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium

[35] **HTML 5**: es la quinta revisión mayor del lenguaje básico de la World Wide Web, HTML. Especifica dos variantes de sintaxis para HTML: un «clásico» HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida como XML (XHTML) (application/xhtml+xml). Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.

[36] **Widget**: es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de *widgets* o Widget Engine. Entre sus objetivos están los de dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.

[37] **X Window**: fue desarrollado a mediados de los años 1980 en el MIT para dotar de una interfaz gráfica a los sistemas Unix. Este protocolo permite la interacción gráfica en red entre un usuario y una o más computadoras haciendo transparente la red para éste. Generalmente se refiere a la versión 11 de este protocolo, X11, el que está en uso actualmente. X es el encargado de mostrar la información gráfica de forma totalmente independiente del sistema operativo.

[38] **Java SE o ME:** Java Platform, Standard Edition y Java Platform, Micro Edition, respectivamente

[39] **Concurso ADC2**: concurso convocado por la plataforma oficial de desarrolladores para seleccionar las mejores aplicaciones de Android.

[40] **Netbook**: Un netbook es un subportátil, es decir, una categoría de ordenador portátil de bajo costo y reducidas dimensiones, lo cual aporta una mayor movilidad y autonomía.

[41] **BitTorrent Azureus**: es un programa para P2P. Es cliente de BitTorrent y es de código abierto. Está desarrollado en lenguaje de programación Java, por lo que es multiplataforma, teniendo instalada la Máquina virtual Java.

[42] **OSGi**: son las siglas de Open Services Gateway Initiative. Su objetivo es definir las especificaciones abiertas de software que permita diseñar plataformas compatibles que puedan proporcionar múltiples servicios. Fue pensado principalmente para su aplicación en redes domésticas y por ende en la llamada Domótica o informatización del hogar.

[43] **DTD**: es una descripción de estructura y sintaxis de un documento XML

[44] **JSP:** es una tecnología Java que permite generar contenido dinámico para Web, en forma de documentos HTML, XML o de otro tipo.

[45] **USB:** es una tecnología Java que permite generar contenido dinámico para Web, en forma de documentos HTML, XML o de otro tipo.

[46] **RDP:** es un protocolo desarrollado por Microsoft que permite la comunicación en la ejecución de una aplicación entre un terminal (mostrando la información procesada que recibe del servidor) y un servidor Windows (recibiendo la información dada por el usuario en el terminal mediante el ratón ó el teclado).

[47] **X86:** es la denominación genérica dada a ciertos microprocesadores de la familia Intel, sus compatibles y la arquitectura básica a la que estos procesadores pertenecen,

[48] **imagen ISO**: es un archivo donde se almacena una copia o imagen exacta de un sistema de ficheros, normalmente un disco óptico.

[49] **Sandbox:** un sistema informático de aislamiento de procesos, mediante el cual, se pueden ejecutar distintos programas con seguridad y de manera separada. A menudo se utiliza para ejecutar código nuevo, o software de dudosa confiabilidad, con objeto de evitar la corrupción de datos del sistema en donde estos se ejecutan. Una maquina Virtual, por ejemplo

[50] **SWT:** es un conjunto de componentes para construir interfaces gráficas en Java, (*widgets*) desarrollados por el proyecto Eclipse.

# 10. Bibliografía

http://www.virtualbox.org/ - Pagina oficial de VirtualBox

http://tomcat.apache.org/ - Pagina oficial de Tomcat

http://www.eclipse.org/ - Pagina oficial de Eclipse

http://www.android-spa.com - Comunidad de desarrolladores en español

http://www.android.com/ - Pagina oficial de Android

http://developer.android.com/ - Pagina oficial para desarrolladores

http://code.google.com/

http://www.anddev.org/- Comunidad de desarrolladores en ingles

http://www.javahispano.com - javaHispano. Tu lenguaje, tu comunidad.

http://programacion.com/java/tutorial/intjava - Java en castellano. Introducción a Java.

http://programacion.com/java/tutorial/servlets_jsp/3 - Java en castellano. Servlets y JSP.

'Servlets y JSP' de Alfonso Cubero y Sergio Luna

*Ilustración 27: Foto a la entrada de Googleplex*

Agradecimientos

A mis padres, por la paciencia, y en especial a Alex y a Rosa por todo el apoyo que me han dado durante el proyecto.